TUGAS AKHIR - KI141502

RANCANG BANGUN MODUL PENGENALAN BAHASA ISYARAT MENGGUNAKAN TEKNOLOGI LEAP MOTION DAN METODE *BACKPROPAGATION-GENETIC ALGORITHM NEURAL NETWORK*

Risal Andika Tridisaputra
NRP 5111100133

Dosen Pembimbing
Wijayanti Nurul Khotimah, S.Kom., M.Sc.
Ridho Rahman Hariadi, S.Kom., M.Sc.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2015

FINAL PROJECT – KI141502

Design and Implementation of Indonesian Sign Language Recognition with Leap Motion Controller and Backpropagation-Genetic Algorithm Neural Network Method

RISAL ANDIKA TRIDISAPUTRA
NRP 5111100133

Advisor
Wijayanti Nurul Khotimah, S.Kom., M.Sc.
Ridho Rahman Hariadi, S.Kom., M.Sc.

INFORMATICS DEPARTMENT
Faculty of Information Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2015

LEMBAR PENGESAHAN

RANCANG BANGUN MODUL PENGENALAN BAHASA ISYARAT MENGGUNAKAN TEKNOLOGI LEAP MOTION DAN METODE *BACKPROPAGATION-GENETIC ALGORITHM NEURAL NETWORK*

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Interaksi Grafis dan Seni
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh:
RISAL ANDIKA TRIDISAPUTRA
NRP. 511110033

Disetujui oleh Pembimbing Tugas Akhir:

1. Wijayanti Nurul Khotimah, S.Kom., M.Sc. ..... (pembimbing 1)
   NIP: 19860312 201212 2 004
2. Ridho Rahman Hariadi, S.Kom., M.Sc. ..... (pembimbing 2)
   NIP: 19870213 201404 1 001

SURABAYA
JUNI, 2015

LEMBAR PENGESAHAN

RANCANG BANGUN MODUL PENGENALAN BAHASA ISYARAT MENGGUNAKAN TEKNOLOGI LEAP MOTION DAN METODE *BACKPROPAGATION-GENETIC ALGORITHM NEURAL NETWORK*

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Interaksi Grafis dan Seni
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh:
RISAL ANDIKA TRIDISAPUTRA
NRP. 511110033

Disetujui oleh Pembimbing Tugas Akhir:

1. Wijayanti Nurul Khotimah, S.Kom., M.Sc. .....
   NIP: 19860312 201212 2 004 (pembimbing 1)

2. Ridho Rahman Hariadi, S.Kom., M.Sc. .....
   NIP: 19870213 201404 1 001 (pembimbing 2)

SURABAYA
JUNI, 2015

RANCANG BANGUN MODUL PENGENALAN BAHASA ISYARAT MENGGUNAKAN TEKNOLOGI LEAP MOTION DAN METODE BACKPROPAGATION-GENETIC ALGORITHM NEURAL NETWORK

Nama Mahasiswa : Risal Andika Tridisaputra
NRP : 5111100133
Jurusan : Teknik Informatika FTIf-ITS
Dosen Pembimbing I : Wijayanti Nurul Khotimah, S.Kom, M.Sc.
Dosen Pembimbing II : Ridho Rahman Hariadi, S.Kom., M.Sc.

ABSTRAK

Bahasa isyarat adalah hal yang penting dalam komunikasi bagi orang yang menderita gangguan pendengaran. Kecepatan menguasai bahasa dan kemampuan mereka berinteraksi sangat dibutuhkan. Mereka membutuhkan bahan pembelajaran yang tidak hanya berisi tentang komponen aural saja, namun juga secara visual karena lebih nyata.

Di sisi lain, teknologi berkembang pesat di segala aspek kehidupan. Berbagai macam terobosan teknologi baru telah diciptakan oleh manusia, salah satunya perangkat Leap Motion yang diciptakan pada sekitar tahun 2013. Dengan menggunakan Leap Motion, manusia dapat melakukan interaksi dengan komputer tanpa sentuhan sama sekali.

Oleh karena itu, penulis memiliki ide yang digunakan dalam tugas akhir ini yaitu akan dibangun sebuah aplikasi pengenalan huruf bahasa isyarat Indonesia sesuai dengan SIBI (Sistem Isyarat Bahasa Indonesia). Tujuannya diharapkan aplikasi ini dapat membantu pengguna belajar tentang bahasa isyarat. Aplikasi ini dapat menerjemahkan bahasa isyarat huruf dengan akurasi di atas 80% jika tangan yang digunakan sama ketika testing. Jika orang yang melakukan testing berbeda, akurasi aplikasi ini sekitar 65-75%.

Kata kunci: Leap Motion, bahasa isyarat, SIBI.

DESIGN AND IMPLEMENTATION OF INDONESIAN SIGN LANGUAGE WITH LEAP MOTION CONTROLLER AND BACKPROPAGATION-GENETIC ALGORITHM NEURAL NETWORK METHOD

Name : **Risal Andika Tridisaputra**
NRP : **5111100133**
Major : **Informatics Department, FTIf-ITS**
Advisor I : **Wijayanti Nurul Khotimah, S.Kom, M.Sc.**
Advisor II : **Ridho Rahman Hariadi, S.Kom., M.Sc.**

ABSTRACT

Sign language is an important thing in the communication of deaf people. The speed of learning sign language and their skills to interact each other is very necessary. They need a media and the learning materials is not only about aural matter, but also visual.

On the other hand, technology growth very quickly. The new Many technology are invented, one of them is Leap Motion which invented in 2013. With Leap Motion Controller, human can interact with computer touchless.

Because of that,the writer has an idea on this final project, it is an application that can be used to learn about Indonesian sign language based on SIBI (Sistem Isyarat Bahasa Indonesia). This application is developed by Microsoft Visual Studio with Leap Motion SDK. The goal of this application is to help people know and learn about Indonesian sign language.The accuracy of this application is more than 80% when the test using the same hand. If it used by another person, the accuracy of this application become 65-75%.

Keywords: : Leap Motion, Sign Language, SIBI

KATA PENGANTAR

Dengan menyebut nama Allah SWT yang Maha Pengasih lagi Maha Panyayang, kami panjatkan puja dan puji syukur atas kehadirat-Nya, yang telah melimpahkan rahmat, hidayah, dan inayah-Nya kepada kami, sehingga penulis dapat menyelesaikan tugas akhir ini dengan baik.

Penulis ingin menyampaikan rasa hormat dan terima kasih yang setinggi-tingginya kepada pihak-pihak yang telah membantu penulis dalam penyelesaian tugas akhir ini, terutama kepada:

1. Bapak Didi Wahyu Rahmadi dan Ibu Nurqolisa Diah Permata, orang tua penulis, yang selalu memberikan dukungan dan semangat baik dalam bantuan finansial maupun dorongan positif agar penulis mampu untuk menyelesaikan tugas akhir dengan benar dan tepat waktu, serta memberikan doa yang terus dikirimkan agar penyelesaian tugas akhir berjalan dengan lancar.
2. Kedua kakak-kakak saya Ryan Ditya Permadi dan Riska Amanda Dwinindita, yang selalu memberikan semangat dan memberikan doa agar penulis mampu menyelesaikan Tugas Akhirnya.
3. Ibu Henning Titi Ciptaningtyas, S.Kom., M.Kom. sebagai dosen wali penulis yang turut memberikan saran dan menuntun penulis dalam menentukan mata kuliah yang akan diambil pada pembelajaran di Jurusan Teknik Informatika ITS ini.
4. Ibu Wijayanti Nurul Khotimah, S.Kom., M.Sc. dan Bapak Ridho Rahman Hariadi, S.Kom., M.Sc. yang telah bersedia untuk menjadi dosen pembimbing tugas akhir sehingga penulis dapat mengerjakan tugas akhir dengan arahan dan bimbingan yang baik dan jelas.
5. Teman-teman Mahasiswa Teknis Informatika 2011 yang telah berjuang bersama-sama selama menempuh pendidikan di Jurusan ini.

6. Teman-teman administrator Lab Komputasi Berbasis Jaringan yaitu Ade, Helmy, Nisa, Deasy, Billa, dan adik-adik yang selalu memberikan inspirasi tersendiri kepada penulis.
7. Teman-teman kontrakan, Rahman, Tommy, Ruslan, Faris, Punggi, Dapik, Andrie, Toto di Wisma Permai yang selalu memberi hiburan kepada penulis.
8. Teman-teman panitia pengurus SCHEMATICS 2013 khususnya Besta, Petrus, Jordy, Baskara, Indra, Ali, Ihe, Risma, Harum, Raras, Uthe dan teman-teman lain yang sudah bersusah payah mengadakan acara untuk mengharumkan jurusan kita.
9. Teman-teman seperjuangan angkatan 2011 Teknik Informatika ITS.
10. Serta pihak-pihak lain yang turut membantu penulis baik secara langsung maupun tidak, yang namanya tidak penulis sebutkan disini.

Penulis telah berusaha sebaik-baiknya dalam menyusun tugas akhir ini, mohon maaf apabila ada kesalahan dan kata-kata yang dapat menyinggung perasaan. Penulis berharap tugas akhir ini dapat menjadi media pembelajaran bahasa isyarat Indonesia.

Surabaya, Juni 2015

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR KODE SUMBER

BIODATA PENULIS

Penulis, Risal Andika Tridisaputra lahir di Jember, Jawa Timur, pada tanggal 9 Juni 1993. Penulis adalah anak ke-3 dari 3 bersaudara dan dibesarkan di Kota Surabaya. Penulis menempuh pendidikan formal di SDN AL-Falah Surabaya (1999-2005), SMPN 32 Surabaya (2005-2008), dan SMAN 2 Surabaya (2008-2011). Pada tahun 2011, penulis menempuh pendidikan S1 jurusan Teknik Informatika Fakultas Teknologi Informasi di Institut Teknologi Sepuluh Nopember, Surabaya, Jawa Timur, Indonesia.

Di jurusan Teknik Informatika, penulis mengambil bidang minat Interaksi Grafis dan Seni dan memiliki kompetensi pada beberapa subjek seperti Desain Web, Pemrograman Android, Pemrograman Windows Phone, dan Big Data. Selama berada di dunia akademi kampus, penulis aktif sebagai asisten dosen untuk mata kuliah Interaksi Manusia dan Komputer. Selain itu penulis juga aktif dalam bidang nonakademik. Organisasi mahasiswa yang pernah diikuti penulis adalah menjadi Staf Departemen Dalam Negeri pada Himpunan Mahasiswa Teknik Computer-Informatika (HMTC) periode 2012-2013, Ketua Sponsorship dalan acara Schematics 2013 yang diselenggarakan oleh Himpunan Mahasiswa Teknik Computer-Informatika (HMTC) periode 2013-2014, dan menjadi peserta di berbagai kegiatan jurusan. Penulis juga pernah menjadi finalis pada beberapa ajang perlombaan seperti, VOCOMFEST tahun 2014, ENUMERATION tahun 2014, dan DISCOVERY tahun 2015. Penulis dapat dihubungi melalui alamat email risal.andika@gmail.com

BAB I
PENDAHULUAN

Pada bab ini akan dipaparkan mengenai garis besar tugas akhir yang meliputi latar belakang, tujuan, rumusan, batasan permasalahan, dan manfaat.

1.1 Latar Belakang

Bahasa isyarat adalah hal yang penting dalam komunikasi bagi orang yang menderita gangguan pendengaran dan juga merupakan cara untuk membaurkan mereka ke dalam masyarakat. Kecepatan menguasai bahasa dan kemampuan mereka berinteraksi sangat dibutuhkan. Mereka membutuhkan suatu media dan bahan pembelajaran tidak hanya berisi tentang komponen aural (berhubungan dengan telinga atau indra pendengaran) saja, namun juga secara visual karena lebih nyata daripada gerakan bibir. Ini merupakan alasan mengapa pengembangan dari bahasa isyarat dibutuhkan [1].

Mengacu pada survei yang dilakukan oleh Multi Center Study di Asia Tenggara, Indonesia berada pada posisi empat teratas dalam jumlah penderita tunarungu didalam populasi negaranya, dengan angka 4,6%, dibandingkan dengan Sri Lanka (8,8%), Myanmar (8,8%) dan India (6,3%) [2]. Pada tahun 1994, Departemen Kebudayaan dan Pendidikan merilis SIBI (Sistem Isyarat Bahasa Indonesia) dalam bentuk kamus. SIBI menjadi bahasa isyarat Indonesia yang resmi. Di dalamnya terdapat posisi jari dan gerakan tangan untuk merepresentasikan kosa kata Bahasa Indonesia. Gerakan isyarat di dalam SIBI telah diatur secara sistematis dan mengikuti konvensi. Namun, media pembelajaran yang beracu pada SIBI belum interaktif karena hanya tersedia dalam bentuk kamus atau buku saja.

Di sisi lain, teknologi informasi masih berkembang pesat di segala aspek kehidupan. Berbagai macam terobosan teknologi baru telah diciptakan oleh manusia, salah satunya perangkat Leap

Motion yang diciptakan pada sekitar tahun 2013. Dengan menggunakan Leap Motion, manusia dapat melakukan interaksi dengan komputer tanpa sentuhan sama sekali.

Leap Motion adalah sebuah alat yang berbentuk seperti perangkat USB *flashdisk*. Hasil deteksi dari alat ini dapat berupa masukan lokasi jari dan tangan dalam bentuk vektor yang bisa diproses selanjutnya oleh komputer. Dengan adanya kemampuan yang dimiliki alat Leap Motion tersebut, muncul sebuah ide untuk membuat rancang bangun modul pengenalan bahasa isyarat menggunakan teknologi Leap Motion. Bahasa isyarat yang digunakan mengacu pada Sistem Isyarat Bahasa Indonesia (SIBI).

1.2 Rumusan Permasalahan

Rumusan masalah yang diangkat dalam tugas akhir ini adalah sebagai berikut:

1. Bagaimana melakukan ekstraksi fitur?
2. Bagaimana membuat *classifier* untuk banyak kelas?
3. Bagaimana membuat aplikasi yang interaktif dan *robust*?

1.3 Batasan Permasalahan

Permasalahan yang dibahas dalam tugas akhir ini memiliki beberapa batasan, di antaranya sebagai berikut:

1. Perangkat lunak berbasis *desktop*.
2. Lingkungan pengembangan aplikasi menggunakan kakas bantu Microsoft Visual Studio.
3. Menggunakan Leap Motion SDK V.2 Skeletal Tracking.
4. Jarak antara Leap Motion Controller dengan ruang *input* tidak lebih dari 25 cm.
5. Algoritma *Neural Network* yang digunakan adalah *Backpropagation-Genetic Algorithm* (BP-GA).
6. Topologi *Neural Network* adalah *Multilayer Perceptron* dengan 1 *hidden layer*.

7. Bahasa isyarat yang digunakan mengacu pada Sistem Isyarat Bahasa Indonesia (SIBI).
8. Bahasa isyarat yang dikenali adalah huruf A sampai Z.
9. Tangan yang dipakai dalam aplikasi ini adalah tangan kanan.

1.4 Tujuan

Tujuan dari pembuatan tugas akhir ini adalah:

1. Dapat merancang bangun modul pengenalan bahasa isyarat dengan menggunakan teknologi Leap Motion.
2. Dapat mengimplementasikan algoritma *Backpropagation* dengan *Genetic Algorithm* untuk proses klasifikasi tangan dari pengguna yang ditangkap oleh Leap Motion.

1.5 Manfaat

Manfaat yang diharapkan dari pengembangan aplikasi tugas akhir ini adalah terciptanya sebuah modul pengenalan bahasa isyarat yang digunakan untuk pembelajaran bagi penderita tunarungu.

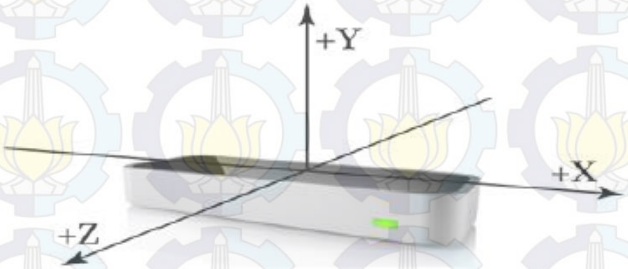
BAB II
DASAR TEORI

Pada bab ini akan dibahas mengenai dasar teori yang menjadi dasar pembuatan tugas akhir ini. Pokok permasalahan yang akan di bahas mengenai teknologi yang mendukung dalam pembuatan tugas akhir seperti Leap Motion SDK, Leap Motion Controller, *neural network*, algoritma *backpropagation*, algoritma *backpropagation-genetic*, dan pengetahuan umum mengenai bahasa isyarat huruf.

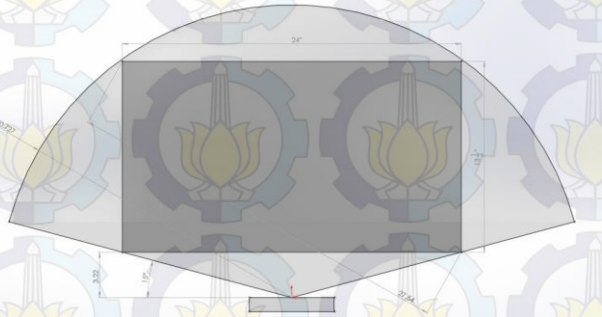
2.1 Leap Motion

Leap Motion adalah alat berukuran kecil berbasis USB yang dapat memungkinkan pengguna untuk melakukan input sebagai masukan komputer tanpa menggunakan sentuhan atau tangan sekalipun. Leap Motion merupakan perangkat keras buatan Leap Motion Inc. dari Amerika. Teknologi Leap Motion dikembangkan pertama pada tahun 2008 oleh David Holz. Pada tahun 2010, alat ini mulai dikenalkan ke publik.

Leap Motion berbentuk seperti perangkat USB *flashdisk* dengan ukuran yang lebih kecil yakni dengan berat 45 gram, panjangnya sekitar 7 sentimeter, lebarnya sekitar 1 sentimeter lebih, dan tingginya sekitar 1 sentimeter lebih seperti pada Gambar 2. 1. Alat ini dapat mendeteksi gerakan tangan dan jari manusia oleh sensor. Alat ini mempunyai 2 inframerah monokrom dan tiga infra merah jenis LED yang mampu mendeteksi area 3 dimensi tepat di atasnya [3]. Jarak deteksi maksimum alat ini sekitar 30 sentimeter dan radius deteksinya berbentuk setengah lingkaran 3 dimensi seperti pada Gambar 2. 2. Alat ini mempunyai presisi 0,7 milimeter terhadap gerakan tangan [4]. Hasil deteksi tersebut dapat berupa masukan lokasi jari dan tangan dalam bentuk vektor yang bisa diproses selanjutnya oleh komputer.



Gambar 2. 1 Leap Motion Controller



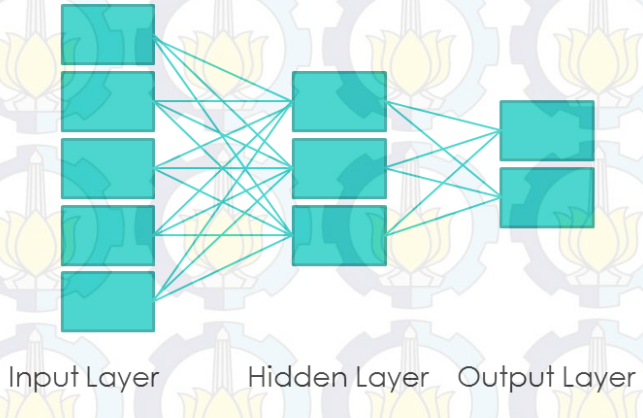
Gambar 2. 2 Radius Deteksi Leap Motion Controller

2.2 Leap Motion SDK

Leap Motion SDK ini dibuat untuk memudahkan pengembang dalam membangun aplikasi. Pengembang dapat mengunduhnya secara gratis dan tersedia dalam berbadai jenis bahasa pemrograman yang berbeda. Antara lain: Javascript, Unity / C#, C++, Java, Python, dan Objective-C. Juga mendukung berbagai macam sistem operasi yang berbeda yakni Windows, OSX, dan Linux [3].

2.3 *Neural Network*

Neural network atau jaringan syaraf neural biasanya terdiri dari 3 bagian yang disebut lapisan atau *layer* [5]. Lapisan *input* terhubung dengan lapisan *hidden* yang juga terhubung dengan lapisan *output*. *Node* yang ada pada lapisan *input* merepresentasikan fitur yang ada pada *network*. *Node* yang ada pada lapisan *hidden* didapatkan dengan aktifitas dari lapisan input dan bobot dari hubungan antara lapisan *input* dan *unit* di lapisan *hidden*. Aktifitas yang terjadi pada lapisan *hidden* tergantung dari aktiftias yang terjadi pada unit di lapisan *hidden* dan bobot antara lapisan *hidden* dan lapisan *output*. Contoh dari bentuk jaringan syaraf dengan 1 lapisan *hidden* dapat dilihat pada Gambar 2. 1.



Gambar 2. 3 Contoh Jaringan Syaraf Feedforward

2.4 *Backpropagation*

Backpropagation (BP) adalah algoritma *training* yang menggunakan *forward network* atau biasanya disebut *Multi Layer Perceptron* (MLP). Jaringan syaraf yang menggunakan algoritma BP memprediksi suatu set tertentu dengan mempelajari pada setiap

contoh yang dimasukkan ke dalam jaringan syaraf [6]. Langkah algoritma BP yaitu:

1. Mencari *error* di *node* ke-j pada lapisan *output* menggunakan rumus pada Persamaan 2.1. *Target* adalah kelas yang diinginkan.

$$Error_j = Output_j(1 - Output_j)(Target_j - Output_j) \quad (2.1)$$

2. Update bobot pada lapisan *hidden* menggunakan rumus pada Persamaan 2.2. *Wij* adalah bobot di antara lapisan i dan j (antara lapisan *hidden* dan *output*).

$$W_{ij} = W_{ij} + (Error_j * Output_i) \quad (2.2)$$

3. Mencari *error* di *node* ke-j pada lapisan *hidden* menggunakan rumus pada Persamaan 2.3. *Node* A terhubung ke *node* B dan *node* C. Error dari *node* B dan *node* C dibutuhkan untuk menghasilkan *error* pada *node* A.

$$Error_a = Output_a(1 - Output_a)(Error_b\, W_{ab} + Error_c\, W_{ac}) \quad (2.3)$$

4. Update bobot pada lapisan *input* menggunakan rumus pada Persamaan 2.2. *Wij* adalah bobot di antara lapisan i dan j (antara lapisan *input* dan *hidden*).

2.5 *Backpropagation-Genetic Algorithm Neural Network*

Sejak Rumelhalt mengenalkan *Backpropagation* (BP) pada tahun 1986, *neural network* BP telah banyak digunakan untuk data

training, dan menjadi bagian penting pada kemajuan *neural network*. Fungsi performa yang digunakan pada BP adalah MSE (*Mean Square Error*).

Rumus yang digunakan untuk mengganti berat dari jaringan pada algoritma BP standar terdapat pada Persamaan 2.4.

$$Wlij(k+1) = Wlij\,(k) + \frac{\mu\,(\partial MSE(w))}{\partial Wlji} \mid w(k) \qquad (2.4)$$

Di mana *k* adalah jumlah iterasi, *Wlij* adalah bobot yang menghubungkan *node* pada layer *i* dan *j*, μ adalah bilangan positif yang disebut *learning rate* yang digunakan untuk mengatur langkah-langkah pembelajaran dan biasanya bernilai bilangan positif sangat kecil.

Algoritma BP adalah metode yang efektif, namun mempunyai beberapa kekurangan yaitu: (1) algoritma BP menggunakan *gradient descent* untuk meminimalisir MSE dimana membuat menjadi *local minimum*; (2) Performa dari algoritma ini sangat sensitif terhadap pengaturan dari *learning rate*. Jika *learning rate* terlalu tinggi, algoritma ini mungkin menjadi tidak stabil. Jika *learning rate* terlalu kecil, algoritma ini membutuhkan waktu untuk menyatu; (3) inisialisasi berat dan bias pada jaringan sangat penting dalam proses konvergensi dari jaringan BP, di mana dihasilkan secara acak, dan terkadang membuat jaringan tidak dapan mencapai tujuan *training*.

Algoritma *Backpropagation – Genetic Algorithm Neural Network* (BPGANN) merupakan algoritma gabungan dari *Back Propagation* dan *Genetic Algorithm* (GA) dengan menggunakan metode GA untuk mencari bobot inisial dari suatu *neural network* dan mempercepat konvergensi, karena algoritma BP membuat konvergensi menjadi pelan. Langkah-langkah algoritma genetik seperti berikut:

1. Menggunakan metode GA untuk mencari kromosom paling unggul sebagai acuan bobot awal pada *neural network*.

2. Menggunakan algoritma BP untuk melatih *neural network* dengan bobot awal yang sudah ada.
3. Jika performa *neural network* semakin berbeda maka lanjut ke langkah 5, jika tidak maka lakukan langkah 4.
4. *Update* bobot dari *neural network* sebagai populasi awal dari GA dan menggunakannya untuk mencari kromosom paling unggul dari bobot yang dilatih oleh algoritma BP, lalu kembali ke langkah nomor 2.
5. Tentukan apakah kondisi pada saat iterasi memuaskan atau tidak, jika sudah memuaskan maka proses pelatihan berhenti, jika tidak maka kembali ke langkah nomor 2 [7].

2.6 Tunarungu

Tunarungu dapat diartikan sebagai keterbatasan yang dimiliki seseorang dalam mendengar sesuatu karena tidak berfungsinya organ pendengaran yang dimilikinya. Ketunarunguan dapat dibedakan menjadi dua kategori yaitu tuli (*deaf*) dan kurang dapat mendengar (*low hearing*) [8]. Tuli adalah keadaan di mana organ pendengaran telah mengalami kerusakan yang sangat parah dan mengakibatkan tidak berfungsinya pendengaran. Sedangkan kurang dapat mendengar adalah keadaan di mana organ pendengaran mengalami kerusakan tetapi masih dapat berfungsi untuk mendengar.

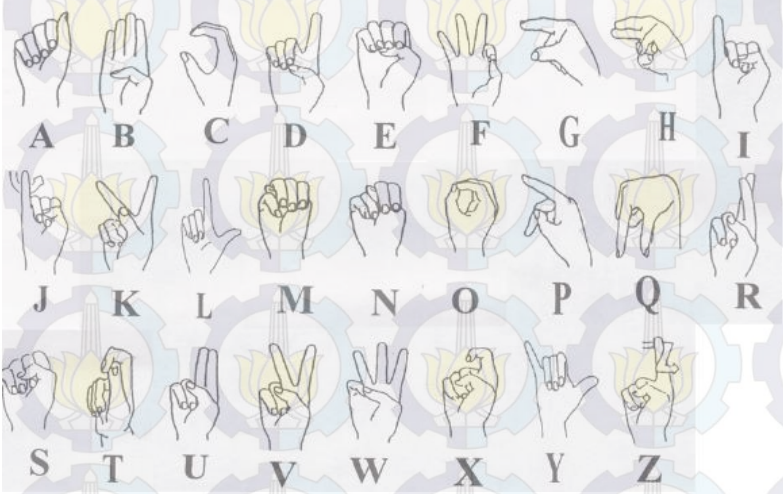
Di Indonesia, terdapat tiga macam sistem komunikasi yang digunakan oleh penderita tunarungu, yaitu:

1. Sistem Oral Murni, meliputi membaca bibir dan berbicara dengan artikulasi jelas.
2. Sistem Isyarat Murni, meliputi kosa kata isyarat, ekspresi wajah dan gerakan anggota tubuh.
3. Sistem Komunikasi Total, meliputi membaca bibir, menulis, isyarat jari, kosa kata isyarat, ekspresi wajah, dan gerakan anggota tubuh.

2.7 Bahasa Isyarat

Penderita tunarungu berkomunikasi dengan orang lain menggunakan bahasa isyarat. Bahasa isyarat berkembang dan memiliki karakteristik sendiri di setiap negara. Di Indonesia, bahasa isyarat yang digunakan mengacu pada kamus Sistem Isyarat Bahasa Indonesia atau biasa disebut dengan SIBI [8]. Jenis isyarat pada SIBI antara lain:

1. Isyarat Pokok adalah isyarat yang melambangkan sebuah kata atau konsep.
2. Isyarat Bentukan adalah isyarat yang dibentuk dengan menggabungkan isyarat pokok dengan isyarat imbuhan.
3. Abjad Jari adalah isyarat yang dibentuk dengan jari-jari untuk mengeja huruf seperti pada Gambar 2. 4.



Gambar 2. 4 Contoh Bahasa Isyarat Huruf

BAB III
ANALISIS DAN PERANCANGAN SISTEM

Bab ini membahas tahap analisis permasalahan dan perancangan dari sistem yang akan dibangun. Analisis permasalahan membahas permasalahan yang diangkat dalam pengerjaan tugas akhir. Analisis kebutuhan mencantumkan kebutuhan-kebutuhan yang diperlukan perangkat lunak. Selanjutnya dibahas mengenai perancangan sistem yang dibuat.

3.1 Analisis Perangkat Lunak

Pada subbab ini akan dibahas mengenai analisa permasalahan dari aplikasi yang akan dibuat, meliputi kebutuhan-kebutuhan perangkat lunak pada aplikasi ini. Hal yang akan dibahas meliputi deskripsi umum perangkat lunak, spesifikasi kebutuhan perangkat lunak, dan identifikasi pengguna.

3.1.1 Deskripsi Umum Perangkat Lunak

Pada tugas akhir ini akan dibangun sebuah aplikasi yang digunakan sebagai alat pengenalan bahasa isyarat yang berpacu pada SIBI (Sistem Isyarat Bahasa Indonesia). Bahasa isyarat yang dikenali adalah bahasa isyarat huruf mulai dari huruf A sampai huruf Z. Sebagai pengganti dari alat *input* perangkat keras seperti *mouse* dan *keyboard*, pada tugas akhir ini penulis akan membuat aplikasi menggunakan Leap Motion. Leap Motion merupakan sebuah alat untuk menangkap sebuah pergerakan (*motion capture*). Alat ini menangkap fitur tangan pengguna sebagai pengganti *input* ke komputer. Lalu, akan dilakukan proses klasifikasi bahasa isyarat dengan algoritma *backpropagation-genetic algorithm*.

Pembangunan aplikasi ini akan menggunakan kakas bantu Microsoft Visual Studio 2010 dengan menggunakan bahasa pemrograman C# dan Leap Motion SDK. Kemudian untuk menggunakannya, pengguna hanya menggunakan tangan sebagai

alat *input* utama, lalu Leap Motion akan mendeteksi koordinat dari tangan untuk ditampilkan ke dalam layar monitor. Arah atas, bawah, kiri, kanan yang digunakan juga sesuai antara layar monitor dan objek. Untuk menekan tombol pada layar cukup menempelkan kursor dengan objek yang ditangkap pada area cakupan Leap Motion.

3.1.2 Spesifikasi Kebutuhan Perangkat Lunak

Kebutuhan sistem yang akan dibuat ini melibatkan dua hal, yakni kebutuhan fungsional maupun kebutuhan non-fungsional. Dimana masing-masing berhubungan dengan keberhasilan dalam pembuatan aplikasi tugas akhir ini.

3.1.2.1 Kebutuhan Fungsional Perangkat Lunak

Pada sistem ini, terdapat beberapa kebutuhan fungsional yang mendukung untuk jalannya aplikasi. Fungsi yang terdapat dalam aplikasi ini adalah sebagai berikut:

a) Deteksi Gerak Objek
   Aplikasi dapat mendeteksi gerakan tangan untuk menekan tombol yang ada pada tampilan antar muka.
b) Mengekstraksi Fitur Tangan
   Aplikasi dapat mendeteksi lokasi objek berupa tangan yang akan diekstraksi menjadi fitur-fitur untuk proses klasifikasi pada saat melakukan *testing*.
c) Menerjemahkan Bahasa Isyarat
   Aplikasi dapat menerjemahkan bahasa isyarat huruf yang sesuai dengan fitur-fitur dari tangan pengguna pada saat melakukan *testing*.

3.1.2.2 Kebutuhan Non-Fungsional

Pada sistem ini, terdapat beberapa kebutuhan non-fungsional yang mendukung dan menambah performa untuk jalannya aplikasi.

Fungsi yang terdapat dalam aplikasi ini adalah sebagai berikut:

a) Penyesuaian Intensitas Cahaya

Intensitas cahaya berpengaruh dalam proses penangkapan gerakan pada Leap Motion. Semakin terang cahaya maka dapat mengganggu sensor pada Leap Motion untuk mendapatkan *input*. Sehingga, pada penggunaan aplikasi ini hendaknya tidak berada pada ruangan yang langsung terkena sinar matahari.

b) Posisi Peletakan Leap Motion

Aplikasi ini menggunakan perangkat keras Leap Motion dengan posisi menghadap pengguna. Posisi peletakan Leap Motion ini disesuaikan dengan penggunaan bahasa isyarat huruf Indonesia pada umumnya.

3.1.3 Identifikasi Pengguna

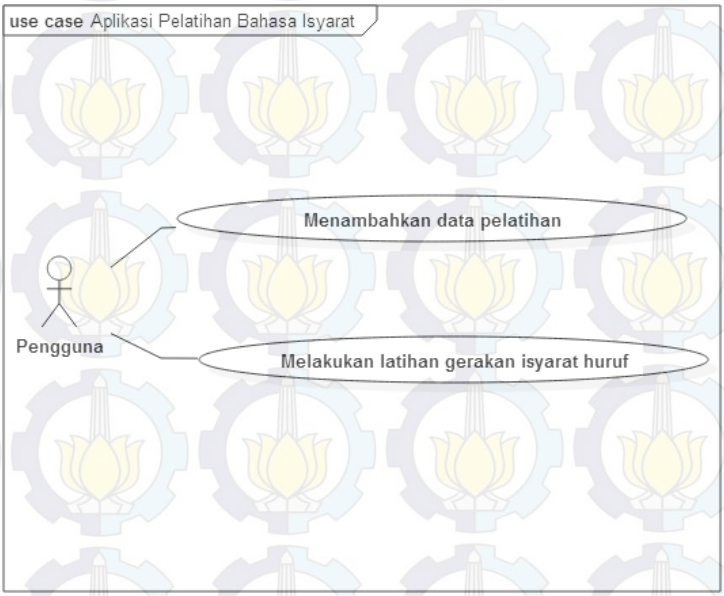
Dalam aplikasi tugas akhir ini, pengguna yang akan terlibat hanya terdapat satu orang saja, yakni orang yang akan melakukan pengenalan bahasa isyarat.

3.2 Perancangan Perangkat Lunak

Subbab ini membahas bagaimana rancangan dari aplikasi tugas akhir ini. Hal yang dibahas meliputi model kasus penggunaan, definisi aktor, definisi kasus penggunaan, arsitektur umum sistem, rancangan antarmuka aplikasi, dan rancangan proses aplikasi.

3.2.1 Model Kasus Penggunaan

Dari hasil analisa deskripsi umum perangkat lunak dan spesifikasi kebutuhan perangkat lunak yang telah dijelaskan, maka model kasus penggunaan untuk aplikasi iSyarat terlihat pada Gambar 3. 1.



Gambar 3. 1 Diagram Kasus Penggunaan Aplikasi

3.2.2 Definisi Aktor

Aktor yang terdapat dalam sistem aplikasi iSyarat terlihat pada Tabel 3. 1.

Tabel 3. 1 Definisi Kasus Penggunaan

No	Nama	Deskripsi
1	Pengguna	Merupakan aktor yang bertugas untuk menambahkan data pelatihan dan melakukan latihan gerakan isyarat huruf, seluruh fungsionalitas yang ada di dalam sistem dapat digunakan oleh pengguna.

3.2.3 Definisi Kasus Penggunaan

Pada Gambar 3. 1 telah dijelaskan bahwa aktor yang dalam hal ini disebut pengguna mempunyai dua kasus penggunaan, yakni menambahkan data pelatihan dan melakukan latihan gerakan isyarat. Detail mengenai kasus penggunaan tersebut dapat dilihat pada Tabel 3. 2.

Tabel 3. 2 Definisi Kasus Penggunaan

No	Kode Kasus Penggunaan	Nama Kasus Penggunaan	Keterangan
1	UC-01	Menambahkan Data Pelatihan	Pengguna dapat menambahkan data pelatihan huruf.
2	UC-02	Melakukan Latihan Gerakan Isyarat Huruf	Pengguna dapat melakukan latihan gerakan isyarat dengan menggunakan data pelatihan yang sebelumnya.

3.2.3.1 Kasus Penggunaan Menambahkan Data Pelatihan

Spesifikasi kasus penggunaan menambahkan data pelatihan dapat dilihat pada Tabel 3. 3.

Tabel 3. 3 Spesifikasi Kasus Penggunaan Menambahkan Data Pelatihan

Nama Kasus Penggunaan	Menambahkan Data Pelatihan
Nomor	UC-01
Deskripsi	Pengguna dapat menambahkan data pelatihan.
Aktor	Pengguna
Kondisi Awal	Pengguna berada pada halaman utama aplikasi.
Alur Normal	1. Halaman utama akan muncul sebagai tanda aplikasi telah dijalankan. 2. Pengguna mengarahkan *pointer* pada layar pada tombol *training*. 3. Pengguna mengarahkan *pointer* pada layar pada tombol mulai. 4. Pengguna menggerakkan jari sesuai dengan bahasa isyarat yang diingkan lalu mengarahkan *pointer* pada layar pada tombol berwarna merah yang terdapat di tengah layar. 5. Pengguna menunggu selama 3 detik. 6. Aplikasi akan menampilkan hasil ekstraksi fitur-fitur tangan dari pengguna. 7. Pengguna memasukkan nama huruf pada *textbox*. 8. Pengguna menekan tombol simpan untuk menyimpan hasil ekstraksi tersebut menjadi sebuah *dataset*.

	9. Pengguna memilih algoritma yang digunakan, dan menginputkan interasi *training*, lalu menekan tombol *training*.
Alur Alternatif	A1. Pengguna menekan tombol *back.* A2. Aplikasi akan kembali menampilkan halaman utama.
Kondisi Akhir	Aplikasi akan menampilkan pemberitahuan bahwa proses pelatihan telah selesai.

3.2.3.2 Kasus Penggunaan Latihan Gerakan Isyarat Huruf

Spesifikasi kasus penggunaan latihan gerakan isyarat huruf dapat dilihat pada Tabel 3. 4.

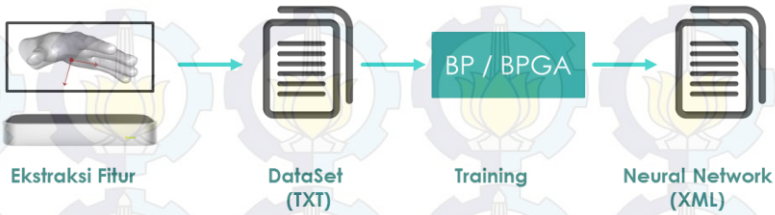
Tabel 3. 4 Spesifikasi Kasus Penggunaan Latihan Gerakan Isyarat Huruf

Nama Kasus Penggunaan	Melakukan Latihan Gerakan Isyarat Huruf
Nomor	UC-02
Deskripsi	Pengguna dapat melakukan latihan gerakan isyarat huruf.
Aktor	Pengguna
Kondisi Awal	Pengguna berada pada halaman utama aplikasi.
Alur Normal	1. Halaman utama akan muncul sebagai tanda aplikasi telah dijalankan. 2. Pengguna mengarahkan *pointer* pada layar pada tombol *testing*. 3. Pengguna mengarahkan *pointer* pada layar pada tombol mulai.

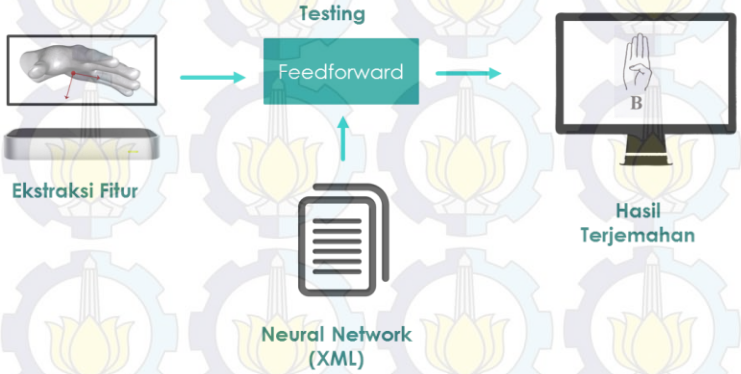
	4. Pengguna menggerakkan jari sesuai dengan bahasa isyarat yang diingkan lalu mengarahkan *pointer* pada layar pada tombol berwarna merah yang terdapat di tengah layar. 5. Pengguna menunggu selama 3 detik.
Alur Alternatif	A1. Pengguna menekan tombol *back*. A2. Aplikasi akan kembali menampilkan halaman utama.
Kondisi Akhir	Aplikasi akan melakukan proses klasifikasi berdasarkan bentuk tangan yang dibuat oleh pengguna pada saat melakukan *testing*.

3.2.4 Arsitektur Umum Sistem

Arsitektur sistem pada aplikasi yang akan dibuat terlihat pada Gambar 3. 2 dan Gambar 3. 3.



Gambar 3. 2 Arsitektur Sistem (*Training*)



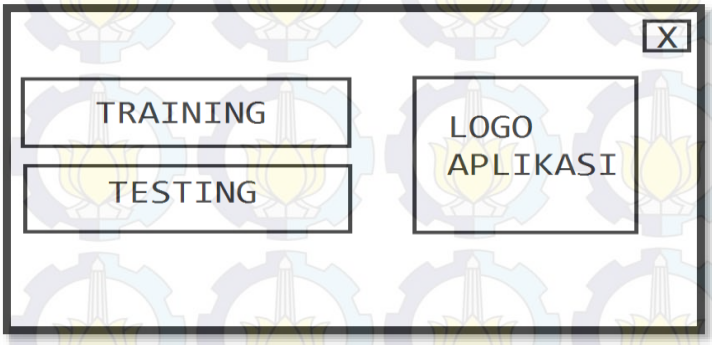
Gambar 3. 3 Arsitektur Sistem (*Testing*)

3.2.5 Rancangan Antarmuka Aplikasi

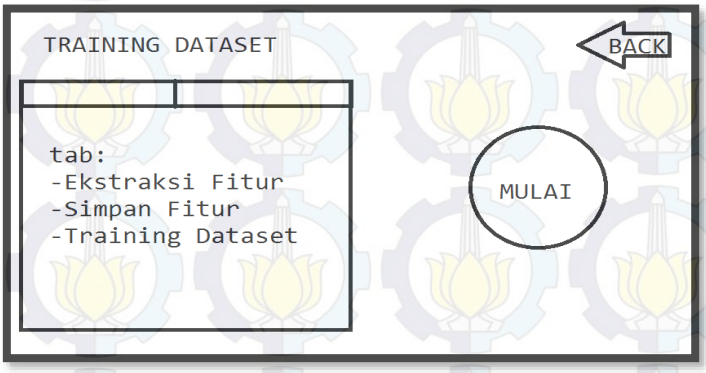
Rancangan antarmuka aplikasi diperlukan untuk memberikan gambaran umum kepada pengguna bagaimana sistem yang ada dalam aplikasi ini berinteraksi dengan pengguna. Selain itu rancangan ini juga memberikan gambaran bagi pengguna apakah tampilan yang sudah disediakan oleh aplikasi mudah untuk dipahami dan digunakan, sehingga akan muncul kesan *user experience* yang baik dan mudah.

3.2.5.1 Rancangan Antarmuka Halaman Utama

Pada halaman ini aplikasi akan menampilkan halaman utama aplikasi yang terdapat 2 tombol yaitu tombol untuk melakukan *testing* dan tombol untuk melakukan *training*. Jika pengguna menggerakkan pointer ke logo aplikasi, maka akan muncul informasi tentang aplikasi ini. Rancangan antarmuka halaman utama bisa dilihat pada Gambar 3. 4.



Gambar 3. 4 Halaman Utama

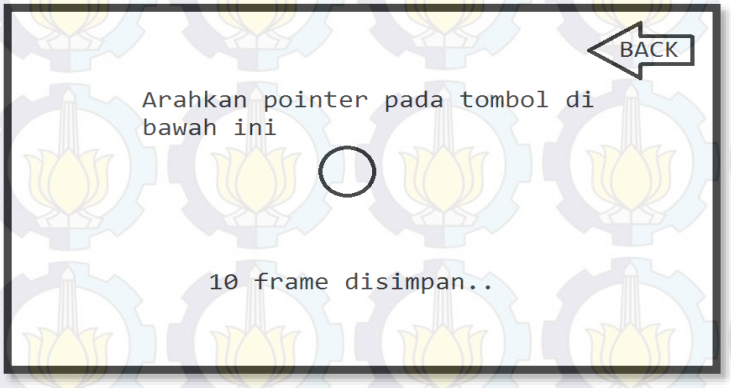


Gambar 3. 5 Halaman *Training*

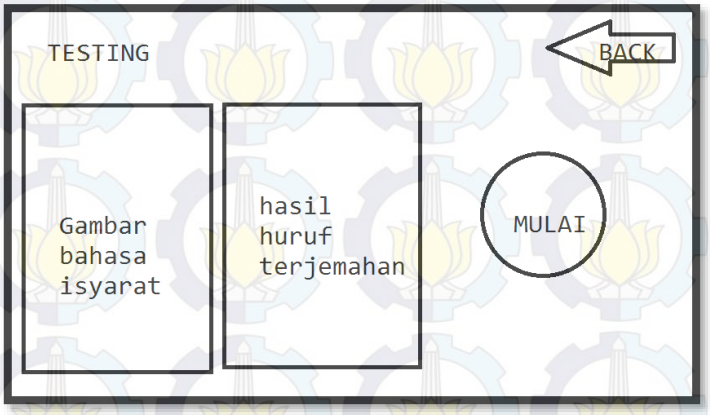
3.2.5.2 Rancangan Antarmuka *Training*

Pada halaman ini aplikasi akan menampilkan halaman *training* yang terdapat tab ekstraksi fitur hasil *training* pengguna, tombol untuk melakukan *training*, tombol untuk menyimpan hasil pelatihan huruf ke dalam bentuk *dataset* dengan ekstensi *.txt, dan tombol untuk menyimpan seluruh *dataset* menjadi sebuah *network* dengan ekstensi *.xml. Rancangan antarmuka halaman *training* bisa dilihat pada Gambar 3. 5.

Setelah pengguna menekan tombol mulai, maka akan muncul halaman pengambilan fitur tangan yang digunakan untuk *training dataset* seperti pada Gambar 3. 6. Setelah fitur disimpan, akan kembali ke halaman *training*.



Gambar 3. 6 Halaman Pengambilan Fitur Tangan Pengguna



Gambar 3. 7 Halaman *Testing*

3.2.5.3 Rancangan Antarmuka *Testing*

Pada halaman ini aplikasi akan menampilkan halaman *testing* yang terdapat hasil terjemahan dari *testing* bahasa isyarat yang dilakukan oleh pengguna. Rancangan antarmuka halaman *testing* dapat dilihat pada Gambar 3. 7.Setelah pengguna menekan tombol mulai, maka akan muncul halaman pengambilan fitur tangan yang digunakan untuk *testing* bahasa isyarat yang dilakukan oleh pengguna seperti pada Gambar 3. 6. Lalu akan dilakukan proses klasifikasi fitur tangan pengguna dan hasilnya akan ditampilkan pada halaman *testing*.

3.2.6 Rancangan Proses Aplikasi

Pada rancangan proses aplikasi akan dijelaskan mengenai proses yang terjadi dalam sistem untuk memenuhi fungsionalitas yang ada pada aplikasi. Proses ini penting agar aplikasi dapat berjalan secara baik dan benar.

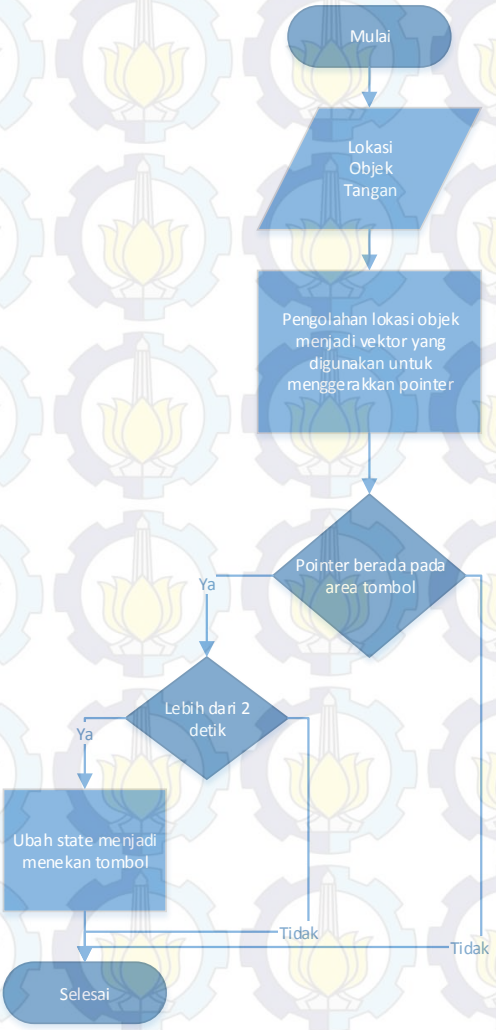
3.2.6.1 Rancangan Proses Pendeteksian Lokasi Objek

Proses ini dibutuhkan sebagai kebutuhan fungsionalitas sistem. Di mana objek yang dimaksud adalah tangan dari pengguna. Lokasi telapak tangan pengguna akan menjadi *pointer* yang dapat bergerak pada layar aplikasi. Diagram alir dapat dilihat pada Gambar 3. 8.

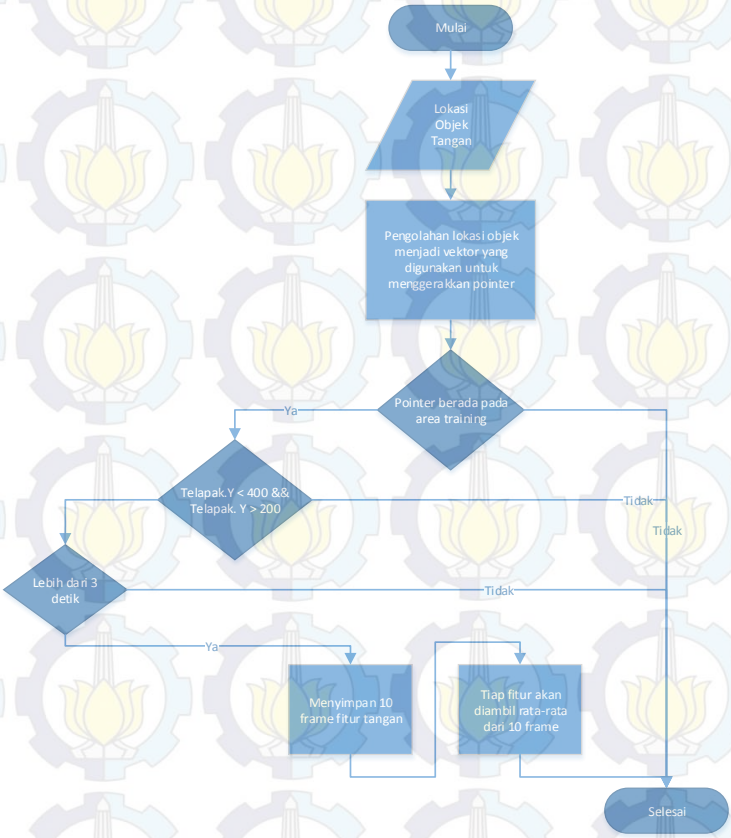
3.2.6.2 Rancangan Proses Ekstraksi Fitur

Proses ekstraksi fitur tangan sangat dibutuhkan bagi pengguna untuk melakukan *training* maupun *testing.* Fitur tangan pengguna disimpan setelah 3 detik. Leap Motion merupakan alat yang sangat sensitif, sehingga fitur tangan yang diambil

menggunakan rata-rata dari 10 *frame* yang terdeteksi. Rancangan proses pengambilan fitur tangan dapat dilihat pada Gambar 3. 9.



Gambar 3. 8 Diagram Alir Mendeteksi Lokasi Objek



Gambar 3. 9 Diagram Alir Ekstraksi Fitur Tangan Pengguna

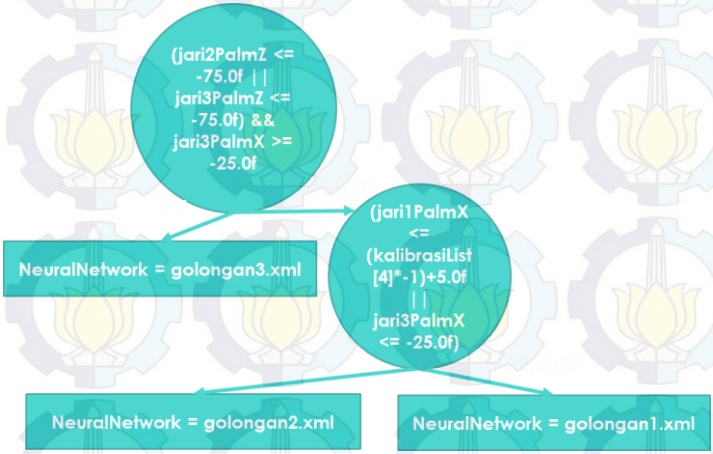
Untuk melakukan ekstraksi tangan pengguna dalam melakukan *training* maupun *testing*, penulis mengambil sebanyak 34 fitur tangan. Ibu jari disebut sebagai jari ke-1, telunjuk disebut sebagai jari ke-2, jari tengah disebut sebagai jari ke-3, jari manis disebut sebagai jari ke-4, dan jari kelingking disebut jari ke-5. Fitur ke-1 sampai dengan 4 merupakan fitur yang diambil dari *paper*. Sedangkan fitur ke-5 sampai 34 merupakan hasil pengamatan yang dilakukan oleh penulis. 34 fitur tangan yang diambil adalah:

1. Radius kepalan tangan.
2. Sudut di antara sumbu z negatif dan proyeksi dari bidang y-z.
3. Sudut di antara sumbu x negatif dan proyeksi dari bidang y-x.
4. Sudut di antara sumbu y negatif dan proyeksi dari bidang x-y.
5. Jarak jari ke-1 dengan telapak tangan terhadap sumbu x.
6. Jarak jari ke-2 dengan telapak tangan terhadap sumbu x.
7. Jarak jari ke-3 dengan telapak tangan terhadap sumbu x.
8. Jarak jari ke-4 dengan telapak tangan terhadap sumbu x.
9. Jarak jari ke-5 dengan telapak tangan terhadap sumbu x.
10. Jarak jari ke-1 dengan telapak tangan terhadap sumbu y.
11. Jarak jari ke-2 dengan telapak tangan terhadap sumbu y.
12. Jarak jari ke-3 dengan telapak tangan terhadap sumbu y.
13. Jarak jari ke-4 dengan telapak tangan terhadap sumbu y.
14. Jarak jari ke-5 dengan telapak tangan terhadap sumbu y.
15. Jarak jari ke-1 dengan telapak tangan terhadap sumbu z.
16. Jarak jari ke-2 dengan telapak tangan terhadap sumbu z.
17. Jarak jari ke-3 dengan telapak tangan terhadap sumbu z.
18. Jarak jari ke-4 dengan telapak tangan terhadap sumbu z.
19. Jarak jari ke-5 dengan telapak tangan terhadap sumbu z.
20. Jarak jari ke-2 dengan jari ke-1 terhadap sumbu x.
21. Jarak jari ke-3 dengan jari ke-1 terhadap sumbu x.
22. Jarak jari ke-4 dengan jari ke-1 terhadap sumbu x.
23. Jarak jari ke-5 dengan jari ke-1 terhadap sumbu x.
24. Jarak jari ke-2 dengan jari ke-1 terhadap sumbu y.
25. Jarak jari ke-3 dengan jari ke-1 terhadap sumbu y.
26. Jarak jari ke-4 dengan jari ke-1 terhadap sumbu y.
27. Jarak jari ke-5 dengan jari ke-1 terhadap sumbu y.
28. Jarak jari ke-2 dengan jari ke-1 terhadap sumbu z.
29. Jarak jari ke-3 dengan jari ke-1 terhadap sumbu z.
30. Jarak jari ke-4 dengan jari ke-1 terhadap sumbu z.
31. Jarak jari ke-5 dengan jari ke-1 terhadap sumbu z.
32. Jarak jari ke-2 dengan jari ke-3 terhadap sumbu x.
33. Jarak jari ke-2 dengan jari ke-3 terhadap sumbu y.
34. Jarak jari ke-2 dengan jari ke-3 terhadap sumbu z.

3.2.6.3 Rancangan Proses Pembagian *Neural Network*

Proses pembagian *neural network* pada aplikasi ini dilakukan dengan membagi 26 huruf menjadi 3 bagian. Karakteristik setiap bagian didapatkan dari hasil pengamatan penulis. Diagram pohon pembagian *neural network* dapat dilihat seperti pada Gambar 3. 10. Sedangkan *Rule* dapat dilihat pada Tabel 3. 5.

Bagian pertama untuk bahasa isyarat huruf yang letak ibu jari pada sumbu x sama atau lebih dari letak jari telunjuk pada sumbu x, yaitu huruf E, I, J, M, N, S, dan T. Bagian kedua untuk bahasa isyarat huruf yang letak ibu jari pada sumbu x lebih kecil dari letak jari telunjuk pada sumbu x, yaitu huruf A, C, D, G, H, O, P, Q, X, dan Y. Bagian ketiga yaitu untuk bahasa isyarat huruf yang jari telunjuk atau jari tengahnya tegak, yaitu huruf B, F, K, L, R, U, V, W, dan Z.



Gambar 3. 10 Diagram Pohon Pembagian *Neural Network*

Tabel 3. 5 *Rule* Pembagian *Neural Network*

Rule	*If*	*Then*
1	(jari2PalmZ <= -75.0f || jari3PalmZ <= -75.0f) && jari3PalmX >= -25.0f	Golongan3.xml
2	(jari1PalmX <= (kalibrasiList[4]*-1)+5.0f || jari3PalmX <= -25.0f)	Golongan2.xml
3	*else*	Golongan1.xml

3.2.6.4 Rancangan Proses Normalisasi Fitur

Normalisasi fitur dilakukan dengan menggunakan Persamaan 3.1. Dilakukan proses *feature scalling* untuk membuat fitur dalam *range* tertentu sehingga fitur lebih proporsional. Nilai maksimum dan minimum setiap fitur didapatkan dari hasil pengamatan yang dilakukan oleh penulis. *Pseudocode* untuk normalisasi fitur seperti pada Gambar 3. 11.

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}} \qquad (3.1)$$

Keterangan:
X = nilai fitur tangan sebelum normalisasi
X' = nilai fitur tangan setelah proses normalisasi
X_{min} = nilai fitur tangan minimum
X_{max} = nilai fitur tangan maksimum.

3.2.6.5 Rancangan Proses Kalibrasi Tangan

Kalibrasi tangan dilakukan dengan cara membandingkan tangan pengguna dengan tangan penulis menggunakan Persamaan 3.2. Setelah mendapatkan *multiplier* (nilai pengali), maka setiap fitur pengguna beracuan pada sumbu tertentu dikalikan dengan

nilai pengali. Proses ini dilakukan karena proses *training* menggunakan tangan penulis. *Pseudocode* untuk mengalibrasi telapak tangan pengguna terdapat pada Gambar 3. 12.

$$M = \frac{N_{pengguna}}{N_{penulis}} \qquad (3.2)$$

Keterangan:
M= nilai pengali
$N_{pengguna}$ = nilai lebar / nilai panjang tangan pengguna
$N_{penulis}$ = nilai lebar / nilai panjang tangan penulis.

Masukan	Fitur tangan pengguna sebelum normalisasi
Keluaran	Fitur tangan pengguna sesudah normalisasi

```
1   for i=0:FiturPengguna.Count()-1
     if i<=3
      Fitur[i] ← Fitur[i]/180
     else if i<=18
      Fitur[i] ← Fitur[i]/100
     else if i<=22
      Fitur[i] ← Fitur[i]/200
     else if i<=26
      Fitur[i] ← Fitur[i]/100
     else if i<=30
      Fitur[i] ← Fitur[i]/200
     else if i<=33
    end
```

Gambar 3. 11 *Pseudocode* Normalisasi Fitur

Masukan	Fitur tangan pengguna sebelum kalibrasi, Fitur tangan penulis yang sudah disimpan
Keluaran	Fitur tangan pengguna sesudah kalibrasi

```
3   MLebar ← (|telapakTangan.X-jariSatu.X|  +
    |telapakTangan.X-jariLima.X|) / LebarJariPenulis
4   MPanjang ← (|telapakTangan.Z-jariTiga.Z|) /
    PanjangJariPenulis
5   for i=0:FiturPengguna.Count()-1
     if i>=4 && i<=9
```

```
    FiturPengguna[i] ← FiturPengguna[i] * MLebar
  else if i>=15 && i<=19
    FiturPengguna[i] ← FiturPengguna[i] * MPanjang
  else if i>=20 && i<=22
    FiturPengguna[i] ← FiturPengguna[i] * MLebar
  else if i>=27 and i<=30
    FiturPengguna[i] ← FiturPengguna[i] * MPanjang
  else if i=31
    FiturPengguna[i] ← FiturPengguna[i] * MLebar
  else if i=33
    FiturPengguna[i] ← FiturPengguna[i] * MPanjang
  end
```

Gambar 3. 12 *Pseudocode* Kalibrasi Tangan Pengguna

BAB IV
IMPLEMENTASI

Bab ini membahas tentang implementasi dari perancangan sistem. Bab ini berisi proses implementasi dari setiap kelas pada semua modul. Namun, pada hasil akhir mungkin saja terjadi perubahan kecil. Bahasa pemrograman yang digunakan adalah bahasa pemrograman C# dengan tambahan menggunakan Leap Motion SDK.

4.1 Lingkungan Pembangunan

Dalam membangun aplikasi ini digunakan beberapa perangkat pendukung baik perangkat keras maupun perangkat lunak. Lingkungan pembangunan dijelaskan sebagai berikut.

4.1.1 Lingkungan Pembangunan Perangkat Keras

Perangkat keras yang digunakan dalam pembuatan aplikasi ini adalah sebuah laptop Lenovo G410S dengan spesifikasi sebagai berikut.

- Prosesor Intel(R) Core i5 CPU @ 2,5GHz
- Memori (RAM) 4,00 GB
- Leap Motion Controller

4.1.2 Lingkungan Pembangunan Perangkat Lunak

Spesifikasi perangkat lunak yang digunakan untuk membuat aplikasi ini sebaga berikut.

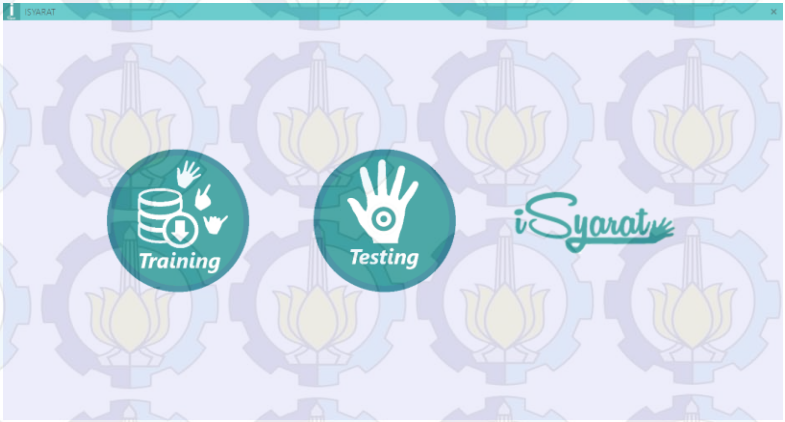
- Microsoft Visual Studio 2012
- Windows 8 64 bit sebagai sistem operasi
- Microsoft Word 2013
- Leap Motion SDK

4.2 Implementasi Antarmuka

Pada subbab ini akan dibahas mengenai hasil implementasi yang dilakukan berdasarkan rancangan antarmuka. Nantinya akan digunakan Leap Motion untuk menggerakkan kursor dalam perpindahan menu dan proses pengenalan bahasa isyarat. Dalam implementasi Antarmuka, digunakan Microsoft Visual Studio 2012 dengan bahasa pemrograman C#.

4.2.1 Implementasi Antarmuka Halaman Utama

Pada halaman utama aplikasi iSyarat ini, terdapat 2 tombol untuk menuju ke menu *training* dan menu *testing*. Cara untuk memilih tombol hanya dengan menggerakkan kursor (menggunakan Leap Motion) ke tombol yang ingin dipilih. Tampilan antarmuka halaman utama seperti pada Gambar 4. 1.

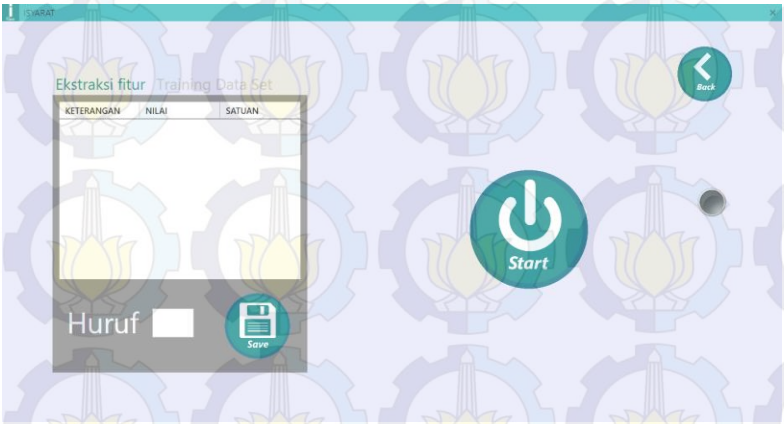


Gambar 4. 1 Antarmuka Halaman Utama

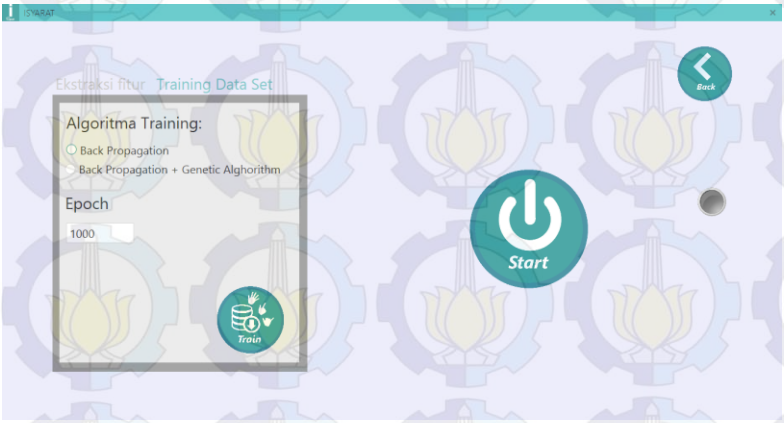
4.2.2 Implementasi Antarmuka Halaman *Training*

Pada halaman *training* ini terdapat tabel ekstraksi fitur, tombol menyimpan *dataset*, tombol untuk mulai merekam fitur

tangan, dan tombol untuk kembali ke halaman sebelumnya yaitu halaman utama seperti pada Gambar 4. 2. Ketika tab "*training dataset*" dipilih, maka akan muncul pilihan untuk melakukan *training* data seperti pilihan algoritma, dan jumlah iterasi yang dilakukan dalam menjalankan algoritma tersebut seperti pada Gambar 4. 3. Jika memilih tombol *start*, maka akan tampil antarmuka seperti pada Gambar 4. 4.



Gambar 4. 2 Antarmuka *Training* (Tabel Ekstraksi Fitur)

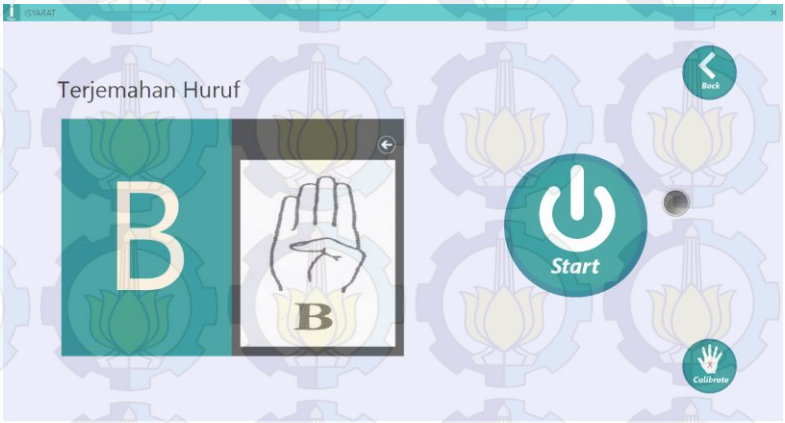


Gambar 4. 3 Antarmuka *Training* (Pilihan *Training* Data)

Gambar 4. 4 Antarmuka *Training* (Proses ekstraksi fitur)

4.2.3 Implementasi Antarmuka Halaman *Testing*

Pada halaman *testing*, terdapat terjemahan dari hasil *testing* beserta gambarnya. Selain itu, juga terdapat tombol untuk memulai *testing*, tombol kembali ke menu sebelumnya, dan juga tombol untuk melakukan kalibrasi. Tampilan antarmuka halaman *testing* dapat dilihat pada Gambar 4. 5.



Gambar 4. 5 Antarmuka *Testing*

Lalu, jika memilih tombol *start*, maka akan tampil halaman pengambilan fitur jari seperti pada Gambar 4. 6. Sedangkan, jika memilih tombol kalibrasi, maka akan tampil halaman untuk kalibrasi seperti pada Gambar 4. 7.

Gambar 4. 6 Antarmuka *Testing* (Proses Ekstraksi Fitur)

Gambar 4. 7 Antarmuka Kalibrasi

4.3 Implementasi Aplikasi

Pada subbab ini akan dibahas mengenai implementasi aplikasi dari kasus penggunaan ke dalam baris kode. Dijelaskan juga dengan fungsi yang dibutuhkan untuk menunjang aplikasi ini agar dapat berjalan sebagaimana mestinya. Implementasi ini dilakukan menggunakan Microsoft Visual Studio 2012 dengan bahasa pemrograman C#.

4.3.1 Implementasi Pendeteksian Lokasi Objek

Untuk menjalankan aplikasi ini tentunya membutuhkan perangkat Leap Motion Controller, sehingga dibutuhkan suatu proses untuk mendeteksi lokasi objek tangan pengguna dan merepresentasikanya kedalam aplikasi ini sebagai kursor.

Kode sumber proses pendeteksian posisi telapak tangan kanan pengguna, dapat dilakukan seperti pada Kode Sumber 4. 1. Sementara kode sumber untuk menggerakkan kursor dapat dilakukan seperti pada Kode Sumber 4. 2

```
void newFrameHandler(Leap.Frame frame, float
deltaTime)
{
 if (frame.Hands.Count != 0){
 leapMouse.Run(frame.Hands.Rightmost.PalmPosition);
 }
}
```

Kode Sumber 4. 1 Kode Sumber Pendeteksian Telapak Tangan

```
public LeapMouse(Image img, MainWindow w){
this.img = img;
this.win = w;
}

public void Run(Vector v){
double velocity = 3.2;
```

```
img.Margin = new Thickness(v.x * velocity +
win.Width / 2, v.z * velocity + win.Height / 2, 0,
0);
}
```

Kode Sumber 4. 2. Kode Sumber Kelas *LeapMouse.cs*

4.3.2. Implementasi Proses Ekstraksi Fitur

Untuk melakukan ekstraksi tangan pengguna dalam melakukan *training* maupun *testing*, penulis mengambil sebanyak 34 fitur tangan. Pada Kode Sumber 4. 3 dijelaskan mengenai proses ekstraksi fitur pada setiap *frame* dari Leap Motion. Setiap fitur akan diambil rata-ratanya dari 10 *frame* tangan yang ditangkap oleh Leap Motion Controller.

```
handSphereRadius += hand.SphereRadius;
handPitch += Math.Abs(hand.Direction.Pitch *
(float)(180.0 / Math.PI));
handYaw += Math.Abs(hand.Direction.Yaw *
(float)(180.0 / Math.PI));
handRoll += Math.Abs(hand.Direction.Roll *
(float)(180.0 / Math.PI));

jari1PalmX += (hand.Fingers[0].TipPosition.x -
hand.PalmPosition.x);
jari2PalmX += (hand.Fingers[1].TipPosition.x -
hand.PalmPosition.x);
jari3PalmX += (hand.Fingers[2].TipPosition.x -
hand.PalmPosition.x);
jari4PalmX += (hand.Fingers[3].TipPosition.x -
hand.PalmPosition.x);
jari5PalmX += (hand.Fingers[4].TipPosition.x -
hand.PalmPosition.x);
jari1PalmY += (hand.Fingers[0].TipPosition.y -
hand.PalmPosition.y);
jari2PalmY += (hand.Fingers[1].TipPosition.y -
hand.PalmPosition.y);
jari3PalmY += (hand.Fingers[2].TipPosition.y -
hand.PalmPosition.y);
```

```
jari4PalmY += (hand.Fingers[3].TipPosition.y -
hand.PalmPosition.y);
jari5PalmY += (hand.Fingers[4].TipPosition.y -
hand.PalmPosition.y);
jari1PalmZ += (hand.Fingers[0].TipPosition.z -
hand.PalmPosition.z);
jari2PalmZ += (hand.Fingers[1].TipPosition.z -
hand.PalmPosition.z);
jari3PalmZ += (hand.Fingers[2].TipPosition.z -
hand.PalmPosition.z);
jari4PalmZ += (hand.Fingers[3].TipPosition.z -
hand.PalmPosition.z);
jari5PalmZ += (hand.Fingers[4].TipPosition.z -
hand.PalmPosition.z);

jari1ke2X += hand.Fingers[1].TipPosition.x -
hand.Fingers[0].TipPosition.x;
jari1ke3X += hand.Fingers[2].TipPosition.x -
hand.Fingers[0].TipPosition.x;
jari1ke4X += hand.Fingers[3].TipPosition.x -
hand.Fingers[0].TipPosition.x;
jari1ke5X += hand.Fingers[4].TipPosition.x -
hand.Fingers[0].TipPosition.x;
jari1ke2Y += hand.Fingers[1].TipPosition.y -
hand.Fingers[0].TipPosition.y;
jari1ke3Y += hand.Fingers[2].TipPosition.y -
hand.Fingers[0].TipPosition.y;
jari1ke4Y += hand.Fingers[3].TipPosition.y -
hand.Fingers[0].TipPosition.y;
jari1ke5Y += hand.Fingers[4].TipPosition.y -
hand.Fingers[0].TipPosition.y;
jari1ke2Z += hand.Fingers[1].TipPosition.z -
hand.Fingers[0].TipPosition.z;
jari1ke3Z += hand.Fingers[2].TipPosition.z -
hand.Fingers[0].TipPosition.z;
jari1ke4Z += hand.Fingers[3].TipPosition.z -
hand.Fingers[0].TipPosition.z;
jari1ke5Z += hand.Fingers[4].TipPosition.z -
hand.Fingers[0].TipPosition.z;
```

```
jari2ke3X += hand.Fingers[2].TipPosition.x -
hand.Fingers[1].TipPosition.x;
jari2ke3Y += hand.Fingers[2].TipPosition.y -
hand.Fingers[1].TipPosition.y;
jari2ke3Z += hand.Fingers[2].TipPosition.z -
hand.Fingers[1].TipPosition.z;
```

Kode Sumber 4. 3 Kode Sumber Ekstraksi Fitur Tangan

4.3.3 Implementasi Algoritma BPGA

Dalam implementasi algoritma *backpropagation*, setiap bobot pada *layer input* maupun bobot pada *layer hidden* akan dideklarasikan terlebih dahulu secara acak. Penulis memberikan *range* angka acak seperti pada Kode Sumber 4. 4.

```
public void InitialiseWeight()
{
 for (int i = 0; i < InputLayer.Count; i++){
 for (int j = 0; j < HiddenLayer.Count; j++){
 InputLayer[i].SetWeight(j, GetRandom() / 100.0f);
  }
 }
 for (int i = 0; i < HiddenLayer.Count; i++){
 for (int j = 0; j < OutputLayer.Count; j++){
   HiddenLayer[i].SetWeight(j, GetRandom() / 100.0f);
  }
 }
}

int GetRandom(){
 return random.Next(0, 30);
}
```

Kode Sumber 4. 4 Fungsi Inisialisasi Bobot Pada Kelas *NeuralNetwork.cs*

Setelah itu, dilakukan proses *feedforward* seperti pada Kode Sumber 4. 5. Tahap algoritma *feedforward* ini adalah:

1. Pada *layer input*, *input layer* ke-i adalah hasil ekstraksi fitur ke-i.
2. Pada *layer hidden*, *input* dari *hidden layer* ke-j dijumlahkan dengan *output* dari *input layer* ke-i (*linear*), lalu lakukan fungsi pengaktifan *sigmoid*.
3. Pada *layer output*, *input* dari *output layer* ke-j dijumlahkan dengan perkalian antara *input hidden layer* ke-i dengan bobot yang acak (*dot product*) [7].

```
public void Run(int index){
 neuralNetwork.InitaliseInput();
 DoInputLayer(index);
}

private void DoInputLayer(int index){
 for (int i = 0; i <
dataSetList[index].AttributeCount; i++){
  neuralNetwork.InputLayer[i].Input =
dataSetList[index][i];
 }
 DoHiddenLayer();
}

private void DoHiddenLayer()
{
 for (int i = 0; i < neuralNetwork.InputLayer.Count;
i++){
 for (int j = 0; j < neuralNetwork.HiddenLayer.Count;
j++){
  neuralNetwork.HiddenLayer[j].Input +=
neuralNetwork.InputLayer[i].GetOutput(j);
  }
 }
 for (int j = 0; j < neuralNetwork.HiddenLayer.Count;
j++){
 neuralNetwork.HiddenLayer[j].Input =
Sigmoid(neuralNetwork.HiddenLayer[j].Input);
  }
  DoOutputLayer();
```

```
}

private void DoOutputLayer(){
for (int i = 0; i < neuralNetwork.HiddenLayer.Count;
i++){
 for (int j = 0; j < neuralNetwork.OutputLayer.Count;
j++)
 {
 neuralNetwork.OutputLayer[j].Input +=
neuralNetwork.HiddenLayer[i].Input *
neuralNetwork.HiddenLayer[i].GetWeight(j);
  }
 }
}

private float Sigmoid(float val){
 return 1 / (1.0f + (float)Math.Exp(-val));
}
```

Kode Sumber 4. 5 Kelas Feedforward.cs

Lalu, akan dilakukan proses *backpropagation*. Tahap algoritma *backpropagation* ini adalah:

1. Menghitung *error* di *output layer* menggunakan rumus pada Persamaan 2.1. Implementasi dapat dilihat pada Kode Sumber 4. 6.
2. *Update* bobot di *hidden layer* menggunakan rumus pada Persamaan 2.2. Implementasi dapat dilihat pada Kode Sumber 4. 8.
3. Menghitung *error* di *hidden layer* menggunakan rumus pada Persamaan 2.3. Implementasi dapat dilihat pada Kode Sumber 4. 9.
4. *Update* bobot di *input layer* menggunakan rumus pada Persamaan 2.2 Implementasi dapat dilihat pada Kode Sumber 4. 10 [7].

Sedangkan algoritma genetik mempunyai tahap-tahap seperti berikut:

1. Deklarasi kromosom seperti pada Kode Sumber 4. 7.
2. Sebanyak iterasi, lakukan *backpropagation* seperti Kode Sumber 4. 13.
3. Lalu, lakukan seleksi kromosom seperti pada Kode Sumber 4. 11.
4. Lalu, lakukan penyilangan kromosom seperti pada Kode Sumber 4. 12.
5. Lalu, lakukan mutasi kromosom seperti pada Kode Sumber 4. 14 [5].

```
private float[] GetErrorOutput(int index){
  float[] outputError = new
float[classificationClass.TargetCount];
  for (int i = 0; i <
classificationClass.TargetCount; i++){
    outputError[i] =
classificationClass.GetTarget(DataSetList[index].Clas
sName)[i] - Network.OutputLayer[i].Input;
  }
  UpdateWeightHidden(outputError, index);
  return outputError;
}
```

Kode Sumber 4. 6 Fungsi GetErrorOutput Pada Kelas Backpropagation.cs

```
private void ChromosomInit()
{
  chromosoms = new List<Chromosom>(individualCount);
  for (int i = 0; i < individualCount; i++){
    chromosoms.Add(new Chromosom(GetRandomBinary()));
  }
}
```

Kode Sumber 4. 7 Fungsi ChromosomInit pada kelas GeneticAlgorithm.cs

```
private void UpdateWeightHidden(float[] outputError,
int index){
for (int i = 0; i < Network.HiddenLayer.Count; i++){
 for (int j = 0; j < Network.OutputLayer.Count; j++){
 float newWeight =
Network.HiddenLayer[i].GetWeight(j) +
(this.learningRate * outputError[j] *
Network.HiddenLayer[i].Input);
Network.HiddenLayer[i].SetWeight(j, newWeight);
  }
 }
GetErrorHidden(outputError, index);
}
```

Kode Sumber 4. 8 Fungsi UpdateWeight di *Hidden Layer* Pada Kelas Backpropagation.cs

```
private void GetErrorHidden(float[] outputError, int
index){
float[] hiddenError = new
float[Network.HiddenLayer.Count];
for (int i = 0; i < Network.HiddenLayer.Count; i++){
 float linear = 0;
 for (int j = 0; j < Network.OutputLayer.Count; j++){
 linear += outputError[j] *
Network.HiddenLayer[i].GetWeight(j);
 }
 hiddenError[i] = Network.HiddenLayer[i].Input * (1 -
Network.HiddenLayer[i].Input) * linear;
 }
 UpdateWeightInput(hiddenError, index);
}
```

Kode Sumber 4. 9 Fungsi GetErrorHidden Pada Kelas Backpropagation.cs

```
private void UpdateWeightInput(float[] hiddenError,
int index){
for (int i = 0; i < Network.InputLayer.Count; i++){
for (int j = 0; j < Network.HiddenLayer.Count; j++){
float newWeight = Network.InputLayer[i].GetWeight(j)
+ (learningRate * hiddenError[j] *
Network.InputLayer[i].Input);
Network.InputLayer[i].SetWeight(j, newWeight);
  }
 }
}
```

Kode Sumber 4. 10 Fungsi UpdateWeightInput Pada Kelas Backpropagation.cs

```
private void DoSelection(){
 chromosoms.Sort((val1, val2) =>
val1.FitnessValue.CompareTo(val2.FitnessValue));
}
```

Kode Sumber 4. 11 Fungsi DoSelection pada kelas GeneticAlgorithm.cs

```
private void DoCrossOver(){
 for (int i = 0; i < chromosoms.Count; i += 2){
 int nextIndex = i + 1;
 if (nextIndex >= chromosoms.Count)
 nextIndex = i - 1;
 int[] bit1 = chromosoms[i].Bit;
 int[] bit2 = chromosoms[nextIndex].Bit;
 for (int j = bit1.Length / 2; j < bit1.Length; j++){
  int t = bit1[j];
  bit1[j] = bit2[j];
  bit2[j] = t;
  }
 chromosoms[i].Bit = bit1;
 chromosoms[nextIndex].Bit = bit2;
 }
}
```

Kode Sumber 4. 12 Fungsi DoCrossOver pada kelas GeneticAlgorithm.cs

```
private void DoBackPropagation(){
  for (int i = 0; i < chromosoms.Count; i++){
  ListDataSet lds = new ListDataSet(listDataSet);
  for (int j = 0; j < lds.Count; j++){
  int popCount = 0;
  for (int k = 0; k < chromosoms[i].Length; k++){
   if (chromosoms[i][k] == 0){
   lds[j].RemoveBit(k - popCount);
   popCount++;
    }
   }
  }
  NeuralNetwork nn = new NeuralNetwork();
  nn.InitialiseNetwork(lds[0].AttributeCount,
lds[0].AttributeCount / 2,
classificationClass.TargetCount);
  nn.InitialiseWeight();

  BackPropagation bp = new BackPropagation();
  bp.Initialise(nn, lds, classificationClass);
  bp.Run(5000);

  FeedForward ff = new FeedForward();
  ff.Initialise(nn, lds);

  int totalCorrect = 0;
  for (int j = 0; j < lds.Count; j++){
    ff.Run(j);
       bool correct = true;
       int[] targetClass =
classificationClass.GetTarget(lds[j].ClassName);
       for (int k = 0; k <
ff.GetActualClass().Length; k++){
        if (targetClass[k] !=
ff.GetActualClass()[k])
         correct = false;
         }
        if (correct)totalCorrect++;
        }
```

```
        chromosoms[i].FitnessValue = totalCorrect /
(float)lds.Count;
        }
}
```

Kode Sumber 4. 13 Fungsi DoBackpropagation pada kelas GeneticAlgorithm.cs

```
private void DoMutation()
{
 for (int i = 0; i < chromosoms.Count; i++){
  int chanceChromoMut = GetRandom();
  if (chanceChromoMut <= GetRandom()){
   int bitIndex = GetRandom(chromosoms[i].Length);
   if (chromosoms[i][bitIndex] == 0)
chromosoms[i][bitIndex] = 1;
   else chromosoms[i][bitIndex] = 0;
  }
 }
}
```

Kode Sumber 4. 14 Fungsi DoMutation pada kelas GeneticAlgorithm.cs

4.3.4 Implementasi Proses Normalisasi Fitur

Proses normalisasi fitur mengimplementasikan *pseudocode* pada Gambar 3. 11, di mana setiap fitur pengguna dibagi dengan nilai maksimum. Nilai maksimum didapatkan penulis pada saat melakukan pengamatan pada setiap fitur. Normalisasi ini dilakukan supaya *training neural network* berjalan dengan cepat. Setiap fitur bernilai antara 0 sampai 1. Implementasi dapat dilihat pada Kode Sumber 4. 15.

4.3.5 Implementasi Proses Kalibrasi Tangan

Proses kalibrasi tangan mengimplementasikan *pseudocode* pada Gambar 3. 12. Kalibrasi tangan dilakukan karena ukuran tangan setiap pengguna berbeda-beda. Implementasi mendapatkan

perbandingan tangan penulis dan pengguna dapat dilihat pada Kode Sumber 4. 16. Sedangkan implementasi proses kalibrasi tangan pengguna dapat dilihat pada Kode Sumber 4. 17.

```
public void Normalized(List<float> kalibrasiList)
{
 for (int i = 0; i < dataSetList.Count; i++)
 {
  for (int j = 0; j < dataSetList[i].AttributeCount;
j++)
  {
  if (j == 0 || j == 1 || j == 2 || j == 3){
   dataSetList[i][j] /= 180.0f;
  }
  else if (j >= 4 && j <= 18){
   dataSetList[i][j] /= 100.0f;
  }
  else if(j>=19 && j<=22){
   dataSetList[i][j] /= 200.0f;
  }
  else if (j >= 23 && j <= 26){
   dataSetList[i][j] /= 100.0f;
  }
  else if (j >= 27 && j <= 30){
   dataSetList[i][j] /= 200.0f;
  }
  else if(j==31){
   dataSetList[i][j] /= 100.0f;
  }
  else if (j == 32){
   dataSetList[i][j] /= 100.0f;
  }
  else if (j == 33){
   dataSetList[i][j] /= 100.0f;
  }
 }
}
```

Kode Sumber 4. 15 Fungsi Normalisasi Fitur

```
if (isCalibratingOnGoing){
 pointable = frame.Pointables.Frontmost;
 hands = frame.Hands;
 hand = hands[0];
 if (countCalibrasi<10){
 kalibrasiLebarTangan +=
Math.Abs(hand.Fingers[0].TipPosition.x -
hand.PalmPosition.x) +
Math.Abs(hand.Fingers[4].TipPosition.x -
hand.PalmPosition.x);
 kalibrasiPanjangTangan +=
Math.Abs(hand.Fingers[2].TipPosition.z -
hand.PalmPosition.z);
 }
 countCalibrasi++;
 if (countCalibrasi == 10){
  kalibrasiPanjangTangan /= 10;
  kalibrasiLebarTangan /= 10;
 }
}
```

Kode Sumber 4. 16 Potongan Kode Sumber Fungsi Mendapatkan Perbandingan Untuk Kalibrasi Tangan

```
public void Calibrated(List<float> multiplier){
 for (int i = 0; i < dataSetList.Count; i++){
  for (int j = 0; j < dataSetList[i].AttributeCount;
j++){
  if (j == 0 || j == 1 || j == 2 || j == 3){
  }
  else if (j == 4 || j == 5 || j == 7 || j == 8 ||
j==9){
   dataSetList[i][j] *= multiplier[35];
  }
  else if (j == 9 || j == 10 || j == 11 || j == 12 ||
j == 13){
  }
  else if (j == 15 || j == 16 || j == 17 || j==18 ||
j==19){
```

```
    dataSetList[i][j] *= multiplier[34];
   }
   else if (j >= 19 && j <= 22){
   dataSetList[i][j] *= multiplier[35];
   }
   else if (j >= 23 && j <= 26){
   }
   else if (j >= 27 && j <= 30){
    dataSetList[i][j] *= multiplier[34];
   }
   else if (j == 31){
    dataSetList[i][j] *= multiplier[35];
   }
   else if (j == 32){
   }
   else if (j == 33){
   dataSetList[i][j] *= multiplier[34];
   }
  }
}
```

Kode Sumber 4. 17 Fungsi Kalibrasi Pada Fitur Pengguna

4.3.6 Implementasi Proses Pembagian *Neural Network*

Proses pembagian *neural network* pada aplikasi ini dilakukan dengan membagi 26 huruf menjadi 3 bagian. Bagian pertama untuk bahasa isyarat huruf yang letak ibu jari pada sumbu x sama atau lebih dari letak jari telunjuk pada sumbu x, yaitu huruf E, I, J, M, N, S, dan T. Bagian kedua untuk bahasa isyarat huruf yang letak ibu jari pada sumbu x lebih kecil dari letak jari telunjuk pada sumbu x, yaitu huruf A, C, D, G, H, O, P, Q, X, dan Y. Bagian ketiga yaitu untuk bahasa isyarat huruf yang jari telunjuk atau jari tengahnya tegak, yaitu huruf B, F, K, L, R, U, V, W, dan Z. Kode sumber untuk proses penggolongan *neural netw*ork dapat dilihat pada Kode Sumber 4. 18.

```
if ((jari2PalmZ <= -75.0f || jari3PalmZ <= -75.0f) &&
jari3PalmX >= -25.0f){
 filename = "gol3_v6_15k.xml";
 sum = 2;
}
else if ((jari1PalmX <= (kalibrasiList[4]*-1)+5.0f ||
jari3PalmX <= -25.0f)){
 filename = "gol2_v6_15k.xml";
 sum = 1;
}
else{
 filename = "gol1_v6_15k.xml";
 sum = 0;
}
```

Kode Sumber 4. 18 Pembagian *Neural Network*

BAB V
PENGUJIAN DAN EVALUASI

Bab ini membahas pengujian dan evaluasi pada aplikasi yang dikembangkan. Pengujian yang dilakukan adalah pengujian terhadap kebutuhan fungsional secara keseluruhan. Pengujian dilakukan dengan beberapa skenario. Hasil evaluasi menjabarkan tentang rangkuman hasil pengujian pada bagian akhir bab ini.

5.1. Lingkungan Pembangunan

Dalam membangun aplikasi ini digunakan beberapa perangkat pendukung baik perangkat keras maupun perangkat lunak. Perangkat keras yang digunakan dalam pembuatan aplikasi ini adalah sebuah laptop Lenovo G410S yang memiliki spesifikasi sebagai berikut.

- Prosesor Intel(R) Core i5 CPU @ 2,50GHz
- Memori (RAM) 4,00 GB
- Leap Motion

5.2. Skenario Pengujian

Skenario pengujian yang dilakukan dibagi menjadi 2 skenario A dan skenario B. Pada skenario A, *neural network* yang digunakan pada saat *testing* adalah hasil *training* dari tangan pengguna yang sama. Sedangkan skenario B, *neural network* yang digunakan pada saat *testing* berbeda dengan pada saat *training*.

1. Pengujian skenario A1 merupakan pengujian akurasi menggunakan algoritma GABP terhadap bahasa isyarat huruf yang dilakukan oleh penulis dengan menggunakan 260 dataset.
2. Pengujian skenario A2 merupakan pengujian akurasi menggunakan algoritma BP terhadap bahasa isyarat huruf

yang dilakukan oleh penulis dengan menggunakan 260 dataset. Pengujian ini menggunakan kalibrasi.

3. Pengujian skenario A3 merupakan pengujian akurasi menggunakan algoritma GABP terhadap bahasa isyarat huruf yang dilakukan oleh penulis dengan menggunakan 260 dataset. Pengujian ini menggunakan kalibrasi.
4. Pengujian skenario A4 merupakan pengujian akurasi terhadap jumlah iterasi yang dilakukan pada algoritma GABP. Pengujian ini dilakukan oleh penulis dengan menggunakan 260 dataset. Pengujian ini menggunakan kalibrasi.
5. Pengujian skenario B1 merupakan pengujian akurasi menggunakan algoritma GABP terhadap bahasa isyarat huruf yang dilakukan oleh tangan orang lain dengan menggunakan 260 dataset. Pengujian ini tanpa menggunakan kalibrasi.
6. Pengujian skenario B2 merupakan pengujian akurasi menggunakan algoritma GABP terhadap bahasa isyarat huruf yang dilakukan oleh tangan orang lain dengan menggunakan 260 dataset. Pengujian ini menggunakan kalibrasi.
7. Pengujian skenario B3 merupakan pengujian akurasi menggunakan algoritma BP terhadap bahasa isyarat huruf yang dilakukan oleh tangan orang lain dengan menggunakan 520 dataset. Pengujian ini menggunakan kalibrasi.
8. Pengujian skenario B4 merupakan pengujian akurasi menggunakan algoritma GABP terhadap bahasa isyarat huruf yang dilakukan oleh tangan orang lain dengan menggunakan 520 dataset. Pengujian ini menggunakan kalibrasi.

5.2.1 Pengujian Skenario A1 dan Analisis

Pada pengujian skenario A1 ini penulis akan melakukan uji coba terhadap akurasi jika testing dilakukan dengan algoritma genetik terlebih dahulu. Uji coba ini menggunakan *dataset* sebanyak 260 buah, semua *dataset* tersebut dilatih oleh tangan penulis. Skenario dapat dilihat pada Tabel 5. 1.

Tabel 5. 1 Skenario Pengujian A1

Nama Skenario Pengujian	Pengujian Akurasi A1
Kode	SP-A1
Algoritma	*Backpropagation-Genetic Algorithm*
Kalibrasi	Tanpa kalibrasi
Jumlah *Dataset*	260 *dataset*
Penguji	Penulis
Prosedur Pengujian	Pengguna melakukan uji coba 26 huruf isyarat sebanyak 5 kali
Hasil yang Diperoleh	Akurasi 86,9%.

Dapat dilihat jika menggunakan algoritma genetik terlebih dahulu untuk menyeleksi fitur tangan yang dipakai, akurasi bertambah sekitar 10%. Pada proses uji coba ini, fitur yang digunakan algoritma genetik pada *neural network* tangan bagian 1 (huruf E, I, J, M, N, S, dan T) adalah fitur 2, 5, 6, 9, 10, 11, 12, 16, 18, 19, 20, 25, 28, 31, 32, dan 34. Sedangkan pada bagian 2 (huruf A, C, D, G, H, O, P, Q, X, dan Y) adalah fitur 1,3,4, 6, 8, 12, 14, 15, 16, 18, 21, 22, 23, 28, 29, 30, dan 31. Sedangkan pada bagian 3 (huruf B, F, K, L, R, U, V, W, dan Z) adalah fitur 2, 3, 4, 5, 6, 7, 9, 12, 13, 14, 15, 16, 19, 21, 25, 28, 31, dan 34. Tabel perbandingan akurasi dapat dilihat pada Tabel 5. 2.

Tabel 5. 2 Tabel Perbandingan Akurasi Algoritma BP dan BPGA

Hasil Uji Coba	**Algoritma**	
	Backpropagation	**BPGA**
Benar	100	113
Salah	30	17
Total Pengujian	130	130
Akurasi	76,90%	86,90%

Menurut analisis yang dilakukan oleh penulis, jika *neural network* yang digunakan menggunakan algoritma genetik terlebih

dahulu, dengan jumlah iterasi yang sama, proses pembuatan *neural network* menggunakan algoritma genetik lebih cepat. Tabel perbandingan kecepatan pembuatan *neural network* dapat dilihat pada Tabel 5. 3.

Tabel 5. 3 Tabel Perbandingan *Runtime* Algoritma (detik)

Neural Network	Algoritma	
	Backpropagation	**BPGA**
Golongan 1	406.56	290.4
Golongan 2	539.8708	385.622
Golongan 3	589.3062	420.933

5.2.2 Pengujian Skenario A2 dan Analisis

Pengujian skenario A2 merupakan pengujian akurasi menggunakan algoritma BP terhadap bahasa isyarat huruf yang dilakukan oleh penulis dengan menggunakan 260 dataset. Pengujian ini menggunakan kalibrasi. Skenario uji coba dapat dilihat pada Tabel 5. 4. Sedangkan hasil uji coba dapat dilihat pada Tabel 5. 5. Pada skenario pengujian A2, huruf J terdeteksi sebagai huruf A sebanyak 5 kali. Hal ini terjadi karena Leap Motion mendeteksi jari kelingking (jari ke-5) sebagai ibu jari (jari ke-1) seperti pada Gambar 5. 1. Selain itu, huruf M terdeteksi sebagai huruf N sebanyak 3 kali. Hal ini terjadi dikarenakan bentuk bahasa isyarat huruf M dan N yang serupa seperti pada Gambar 5. 2. Selain itu, dengan menggunakan kalibrasi, akurasi bertambah sekitar 4%.

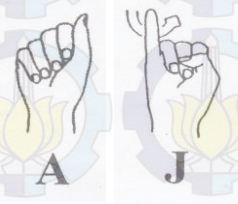
Tabel 5. 4 Skenario Pengujian A2

Nama Skenario Pengujian	Pengujian Akurasi A2
Kode	SP-A2
Algoritma	*Backpropagation*
Kalibrasi	Kalibrasi

Jumlah *Dataset*	260 *dataset*
Penguji	Penulis
Prosedur Pengujian	Pengguna melakukan uji coba 26 huruf isyarat sebanyak 5 kali
Hasil yang Diperoleh	Akurasi 80,80%.

Tabel 5. 5 Terjemah Huruf Isyarat Skenario Pengujian A2

Hasil Uji Coba \ Target Kelas	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	5									5																
b		5																								
c			5																							
d				5																				2		
e					4																					
f						5																				
g							5																			
h								5																		
i									4																	
j					1					0																
k											3											1				
l												5														
m									1				1													
n													3	4					1							
o															5											
p																4	1									
q																	4									
r																		3			1					1
s													1	1					4	1						
t																				4						
u																		2			4					
v											2											4				
w																							5			
x																1								3		
y																									5	
z																										4



Gambar 5. 1 Perbandingan Huruf A dan J

Gambar 5. 2 Perbandingan Huruf M dan N

5.2.3 Pengujian Skenario A3 dan Analisis

Pengujian skenario A3 merupakan pengujian akurasi menggunakan algoritma GABP terhadap bahasa isyarat huruf yang dilakukan oleh penulis dengan menggunakan 260 dataset. Pengujian ini menggunakan kalibrasi. Pada skenario pengujian A3, akurasi terjemah bahasa isyarat naik 13% dari yang semula 80,80% menjadi 93,80%. Skenario uji coba dapat dilihat pada Tabel 5. 6. Sedangkan hasil uji coba dapat dilihat pada Tabel 5. 7.

Tabel 5. 6 Skenario Pengujian A3

Nama Skenario Pengujian	Pengujian Akurasi *Testing* A3
Kode	SP-A3
Algoritma	*Backpropagation-Genetic Algorithm*
Kalibrasi	Kalibrasi
Jumlah *Dataset*	260 *dataset*
Penguji	Penulis
Prosedur Pengujian	Pengguna melakukan uji coba 26 huruf isyarat sebanyak 5 kali
Hasil yang Diperoleh	Akurasi 93,80%.

5.2.4 Pengujian Skenario A4 dan Analisis

Pengujian skenario A4 merupakan pengujian akurasi menggunakan algoritma GABP terhadap iterasi yang dilakukan

dengan menggunakan 260 dataset. Pengujian ini menggunakan kalibrasi. Tabel perbandingan akurasi dapat dilihat pada Tabel 5. 8. Pada pengujian skenario A4 ini, didapatkan hasil iterasi yang paling optimal adalah ketika iterasi *backpropagation* yang dilakukan saat sudah mendapatkan kromosom yang baik dengan algoritma genetika adalah pada jumlah iterasi 15000.

Tabel 5. 7 Terjemah Huruf Isyarat Pengujian Skenario A3

Hasil Uji Coba \ Target Kelas	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	5																							2		
b		5																								
c			5																							
d				5												1										
e					5																					
f						5																				
g							5																			
h								5																		
i									5																	
j										5																
k											5															
l												5														
m													4													
n													1	4												
o															5											
p																4										
q																	5									
r																		3								1
s														1					5							
t																				5						
u																		2			5					
v																						5				
w						1																		5		
x																									3	
y																										5
z																										4

Tabel 5. 8 Perbandingan Akurasi Terhadap Jumlah Iterasi

Jumlah Iterasi	Akurasi
10000	83,84 %
15000	93,80 %
20000	87,60 %

5.2.5 Pengujian Skenario B1 dan Analisis

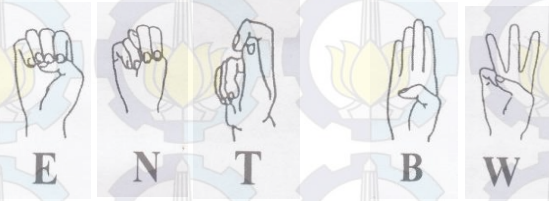
Pada pengujian skenario B1 ini, penulis akan melakukan uji coba terhadap akurasi yang didapatkan pada saat melakukan *testing* yang dilakukan oleh orang lain (panjang jari tengah dari telapak tangan adalah 91.5 mm dan lebar telapak tangan 51.6 mm) menggunakan algoritma *backpropagation-genetic algorithm*. Uji coba ini menggunakan *dataset* sebanyak 260 buah, semua *dataset* tersebut dilatih oleh tangan penulis. Skenario dapat dilihat pada Tabel 5. 9. Sedangkan hasil uji coba akurasi setiap huruf dapat dilihat seperti pada Tabel 5. 10.

Tabel 5. 9 Skenario Pengujian B1

Nama Skenario Pengujian	Pengujian Akurasi B1
Kode	SP-B1
Algoritma	*Backpropagation-Genetic Algorithm*
Kalibrasi	Tanpa kalibrasi
Jumlah *Dataset*	260 *dataset*
Penguji	Pengguna dengan panjang jari tengah 91.5 mm dan lebar telapak tangan 51.6 mm
Prosedur Pengujian	Pengguna melakukan uji coba 26 huruf isyarat sebanyak 5 kali
Hasil yang Diperoleh	Akurasi 66,9%.

Dari hasil pengujian ini, akurasi hanya mencapai 66,9%. Menurut analisis yang dilakukan oleh penulis, penyebab dari kecilnya akurasi tersebut dikarenakan tangan pengguna yang lebar dan panjang telapak tangannya lebih kecil dari tangan penulis dan tanpa melakukan kalibrasi terlebih dahulu. Huruf E terdeteksi sebagai huruf N sebanyak 3 kali juga terdeteksi sebagai huruf T sebanyak 2 kali. Huruf B terdeteksi sebagai huruf W sebanyak 3

kali. Selain bentuk tangan yang tidak sama, huruf yang dilakukan uji coba juga serupa seperti pada Gambar 5. 3.



Gambar 5. 3 Perbandingan Huruf Skenario Pengujian B1

Tabel 5. 10 Terjemah Huruf Isyarat Pengujian Skenario B1

Hasil Uji Coba	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	2																									
b		2				1																				2
c			4																							
d				4																						
e	1				0								1													
f						4																				
g							2																			
h								5																		
i									4																	
j										5									1							
k											2															
l												0														
m													1						1		2					
n					3								3	5						1						
o			1	1											5											
p																5								2		
q						3											5									
r																		5								2
s																			1							
t	2				2				1										2	4						
u																					2					
v											3											5				
w		3										5									1		5			
x																								3		
y																									5	
z																										2

5.2.6 Pengujian Skenario B2 dan Analisis

Pada pengujian skenario B2 ini, penulis akan melakukan uji coba terhadap akurasi yang didapatkan pada saat melakukan *testing* yang dilakukan oleh orang lain (panjang jari tengah dari telapak tangan adalah 91.5 mm dan lebar telapak tangan 51.6 mm) menggunakan algoritma *backpropagation-genetic algorithm*. Uji coba ini menggunakan *dataset* sebanyak 260 buah, semua *dataset* tersebut dilatih oleh tangan penulis. Uji coba ini dilakukan dengan kalibrasi terlebih dahulu. Skenario dapat dilihat pada Tabel 5. 12. Sedangkan hasil uji coba akurasi setiap huruf dapat dilihat seperti pada Tabel 5. 11.

Tabel 5. 11 Tabel Terjemah Skenario Pengujian B2

Hasil Uji Coba \ Target Kelas	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	4																									
b		3				1																				1
c			3																							
d				3																						
e	1				0								1													
f						4																				
g							3																			
h								5																		
i									3																	
j										5									1							
k											2															
l												3														
m													1						1							
n					3								3	5						1						
o			2	2											5											
p																5								2		
q							2										5									
r																		5								2
s																			1		2					
t					2				2										2	4						
u																					3					
v											3											5				
w		2										2											5			
x																								3		
y																									5	
z																										2

Tabel 5. 12 Skenario Pengujian B2

Nama Skenario Pengujian	Pengujian Akurasi B2
Kode	SP-B2
Algoritma	*Backpropagation-Genetic Algorithm*
Kalibrasi	Kalibrasi
Jumlah *Dataset*	260 *dataset*
Penguji	Pengguna dengan panjang jari tengah 91.5 mm dan lebar telapak tangan 51.6 mm
Prosedur Pengujian	Pengguna melakukan uji coba 26 huruf isyarat sebanyak 5 kali
Hasil yang Diperoleh	Akurasi 70,80%.

Pada pengujian ini, dapat dilihat bahwa pada *neural network* golongan 1 (huruf E, I, J, M, N, S) sering terjadi salah terjemah. Menurut analisa penulis, hal ini dikarenakan fitur ke 21, 22, dan 23 (jarak jari ke-1 dengan jari ke 3, 4, dan 5 pada sumbu X) diseleksi. Namun, akurasi bertambah sekitar 3,9% setelah dilakukan kalibrasi.

5.2.7 Pengujian Skenario B3 dan Evaluasi

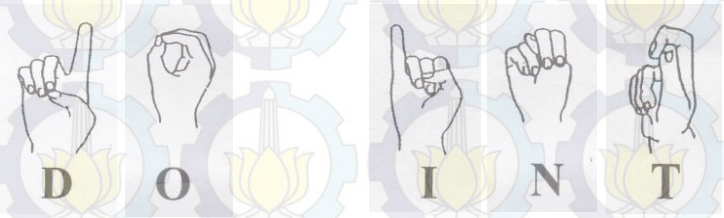
Pada pengujian skenario B3 ini, penulis akan melakukan uji coba terhadap akurasi yang didapatkan pada saat melakukan *testing*. *Testing* ini dilakukan oleh orang lain (panjang jari tengah dari telapak tangan adalah 110,56 mm dan lebar telapak tangan 74.3 mm). Uji coba ini menggunakan *dataset* sebanyak 520 buah, semua *dataset* tersebut dilatih oleh tangan penulis. Uji coba ini dilakukan dengan mengalibrasi tangan pengguna terlebih dahulu. Skenario dapat dilihat pada Tabel 5. 14. Sedangkan hasil uji coba akurasi setiap huruf dapat dilihat seperti pada Tabel 5. 13

Tabel 5. 14 Skenario Pengujian B3

Nama Skenario Pengujian	Pengujian Akurasi B3
Kode	SP-B3
Algoritma	*Backpropagation*
Kalibrasi	Kalibrasi
Jumlah *Dataset*	520 *dataset*
Penguji	Pengguna dengan panjang jari tengah 91.5 mm dan lebar telapak tangan 51.6 mm
Prosedur Pengujian	Pengguna melakukan uji coba 26 huruf isyarat sebanyak 5 kali
Hasil yang Diperoleh	Akurasi 68,5 %.

Tabel 5. 13 Tabel Terjemahan Huruf Skenario B3

Hasil Uji Coba \ Target Kelas	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	4									5																
b		5																								
c			5																							
d				1																						
e					5																					
f						5																				
g							5																			
h								1																		
i									1																	
j										0																
k											1															
l												5														
m													0							1						
n									2				5	0					2							
o	1			4				4							5									5		
p																5										
q																	5									
r																		5								
s														5					2							
t								2											1	4						
u																					5					
v										4												5				
w																							5			
x																								0		
y																									5	
z																										5



Gambar 5. 4 Perbandingan Huruf Skenario Pengujian B3

Pada skenario pengujian B3, huruf I terdeteksi sebagai huruf J sebanyak 5 kali, huruf J terdeteksi sebagai huruf I sebanyak 5 kali. Hal ini terjadi karena Leap Motion mendeteksi jari kelingking pengguna sebagai ibu jari pada saat melakukan isyarat huruf I dan J seperti pada Gambar 5. 5. Selain itu, huruf D terdeteksi sebagai huruf O sebanyak 4 kali, huruf I terdeteksi sebagai huruf N sebanyak 2 kali dan huruf T sebanyak 2 kali, huruf K terdeteksi sebagai huruf V sebanyak 4 kali, Huruf S terdeteksi sebagai huruf N sebanyak 2 kali dan huruf T sebanyak 1 kali, dan huruf X terdeteksi sebagai huruf O sebanyak 5 kali. Menurut analisis yang dilakukan oleh penulis, hal ini terjadi karena huruf yang dilakukan *testing* serupa seperti pada Gambar 5. 4.

5.2.8 Pengujian Skenario B4 dan Evaluasi

Pada pengujian skenario B4 ini penulis akan melakukan uji coba terhadap akurasi yang didapatkan pada saat melakukan *testing*. *Testing* ini dilakukan oleh orang lain (panjang jari tengah dari telapak tangan adalah 110,56 mm dan lebar telapak tangan 74.3 mm). Uji coba ini menggunakan *dataset* sebanyak 520 buah, semua *dataset* tersebut dilatih oleh tangan penulis. Uji coba ini dilakukan dengan mengalibrasi tangan pengguna terlebih dahulu. Skenario dapat dilihat pada Tabel 5. 16. Sedangkan hasil uji coba akurasi setiap huruf dapat dilihat seperti pada Tabel 5. 15.

Tabel 5. 15 Terjemah Huruf Isyarat Pengujian Skenario B4

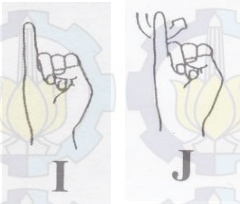
Hasil Uji Coba \ Target Kelas	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	5			2						5																
b		5																								
c			5																							
d				3																						
e					5																					
f						5																				
g							5																			
h								3																		
i									0																	
j									5	0																
k											1											1				
l												5														
m													0													
n													4	5					2	3						
o															5											
p								2								5										
q																	5									
r																		3								
s													1						3							
t																				2						
u																		2			5					
v											4											4				
w																							5			
x																								5		
y																									5	
z																										5

Tabel 5. 16 Skenario Pengujian B4

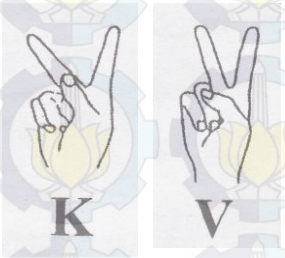
Nama Skenario Pengujian	Pengujian Akurasi B4
Kode	SP-B4
Algoritma	*Backpropagation-Genetic Algorithm*
Kalibrasi	Kalibrasi
Jumlah *Dataset*	520 *dataset*

Penguji	Orang lain dengan panjang jari tengah 110,56 mm dan lebar telapak tangan 74.3 mm
Prosedur Pengujian	Pengguna melakukan uji coba 26 huruf isyarat sebanyak 5 kali
Hasil yang Diperoleh	Akurasi 76,15%.

Pada skenario pengujian B4, huruf I terdeteksi sebagai huruf J sebanyak 5 kali, huruf J terdeteksi sebagai huruf I sebanyak 5 kali. Hal ini terjadi karena Leap Motion mendeteksi jari kelingking pengguna sebagai ibu jari pada saat melakukan isyarat huruf I dan J seperti pada Gambar 5. 5. Selain itu, huruf K terdeteksi sebagai huruf V sebanyak 4 kali. Hal ini terjadi dikarenakan bentuk bahasa isyarat huruf K dan huruf V yang serupa seperti pada Gambar 5. 6. Namun, ketika melakukan uji coba bahasa isyarat V, hasil terjemah tidak terdeteksi sebagai huruf K.



Gambar 5. 5 Perbandingan Huruf I dan J



Gambar 5. 6 Perbandingan Huruf K dan V

5.3. Evaluasi

Pada subbab ini, akan dijelaskan mengenai perbandingan dalam bentuk grafik terhadap uji coba yang telah dilakukan sebelumnya. Beberapa parameter perbandingan diantara lain:

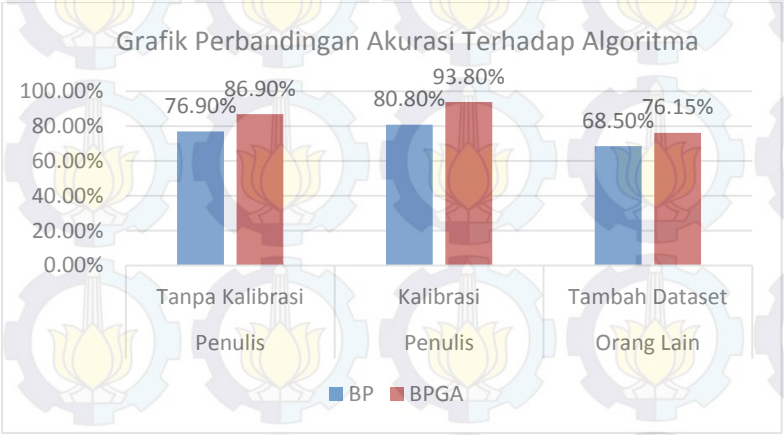
1. Akurasi model *neural network* Terhadap algoritma yang dipakai
2. Akurasi model jika ada penambahan *dataset.*
3. Akurasi model terhadap proses kalibrasi.
4. Akurasi model terhadap iterasi yang dilakukan.
5. *Runtime* model *neural network* Terhadap algoritma yang dipakai.

5.3.1 Perbandingan Akurasi Model *Neural Network* Terhadap Algoritma yang Dipakai

Pada pengujian skenario A1, selisih akurasi yang didapatkan *neural network* menggunakan algoritma genetik dan tanpa menggunakan algoritma genetik adalah sekitar 10%. Grafik perbandingan akurasi dari skenario A1 dapat dilihat seperti pada Gambar 5. 7. Selain itu, jika pengujian dilakukan kalibrasi tangan terlebih dahulu seperti pada skenario A2 dan A3, selisih akurasi sekitar 13%. Grafik perbandingan akurasi dari skenario A2 dan A3 dapat dilihat seperti pada Gambar 5. 7.

5.3.2 Perbandingan Akurasi Model Jika Ada Penambahan *Dataset*

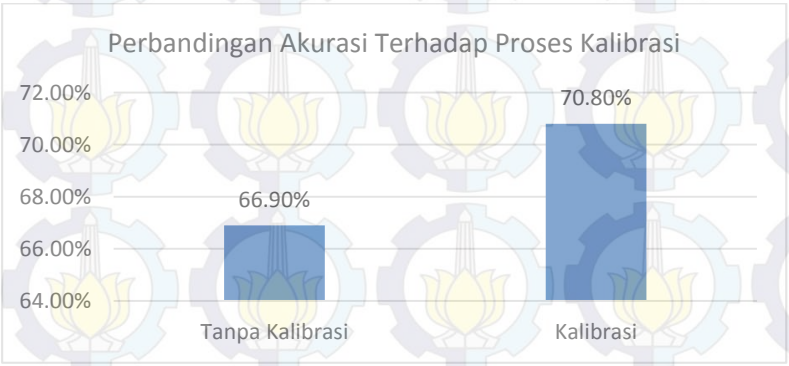
Pada pengujian skenario B3 dan B4, selisih akurasi yang didapatkan *neural network* menggunakan algoritma genetik dan tanpa menggunakan algoritma genetik adalah sekitar 7,65 %. Penambahan *dataset* dilakukan dengan tujuan untuk memperkaya model pada *neural network*. Grafik perbandingan akurasi dapat dilihat pada Gambar 5. 7.



Gambar 5. 7 Grafik Perbandingan Terhadap Algoritma

5.3.3 Perbandingan Akurasi Model Jika Menggunakan Kalibrasi

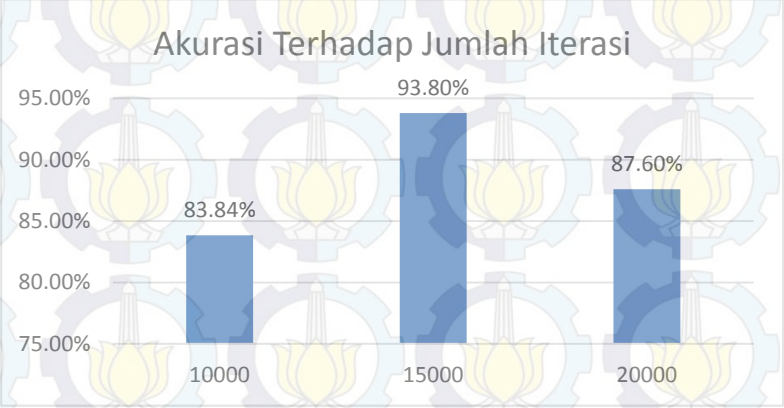
Pada pengujian skenario B1 dan B2, selisih akurasi yang didapatkan *neural network* tanpa menggunakan kalibrasi dan menggunakan kalibrasi terlebih dahulu adalah sekitar 3,9 %. Grafik perbandingan akurasi dapat dilihat pada Gambar 5. 8.



Gambar 5. 8 Grafik Perbandingan Terhadap Proses Kalibrasi

5.3.4 Perbandingan Akurasi Model Terhadap Jumlah Iterasi

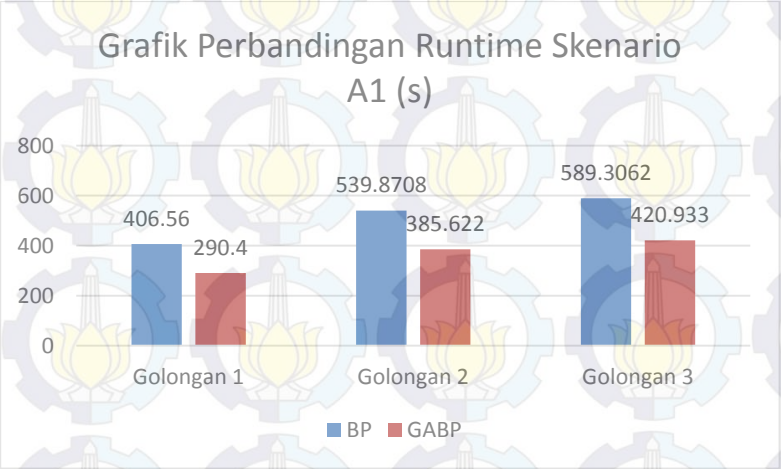
Pada pengujian skenario A4, didapatkan hasil iterasi yang paling optimal adalah ketika iterasi *backpropagation* yang dilakukan saat sudah mendapatkan kromosom yang baik dengan algoritma genetika adalah pada jumlah iterasi 15000 yaitu mencapai 93,80%. Perbandingan akurasi dapat dilihat pada Gambar 5. 9.



Gambar 5. 9 Grafik Perbandingan Akurasi Terhadap Jumlah Iterasi

5.3.5 Perbandingan *Runtime* Aplikasi Model *Neural Network* Terhadap Algoritma yang Dipakai

Pada pengujian skenario A1, *runtime* pada saat melakukan training pada *neural network* menggunakan algoritma genetika lebih cepat daripada tanpa menggunakan algoritma genetika terlebih dahulu. Grafik perbandingan *runtime* dapat dilihat pada Gambar 5. 10.



Gambar 5. 10 Grafik Perbandingan *Runtime* Skenario A1

BAB VI
KESIMPULAN DAN SARAN

Pada bab ini akan diberikan kesimpulan yang diambil selama pengerjaan tugas akhir serta saran-saran tentang pengembangan yang dapat dilakukan terhadap tugas akhir ini di masa yang akan datang.

6.1 Kesimpulan

Dari proses pengerjaan selama perancangan, implementasi, dan proses pengujian aplikasi yang dilakukan, dapat diambil kesimpulan sebagai berikut.

1. Dari sisi akurasi, penerapan algoritma GABP pada aplikasi ini lebih akurat daripada algoritma BP. Algoritma GABP dapat meningkatkan akurasi *neural network* sekitar 7-13%.
2. Dari sisi *runtime*, penerapan algoritma GABP pada aplikasi ini lebih cepat daripada algoritma BP.
3. Aplikasi yang dibangun pada tugas akhir ini dapat menerjemahkan bahasa isyarat huruf dengan akurasi di atas 80% jika pada saat *testing* menggunakan data *training* tangan yang sama. Namun, tangan yang ada pada data *training* berbeda pada saat *testing*, akurasi aplikasi ini menjadi sekitar 65-75%.
4. Leap Motion merupakan alat yang sangat sensitif terhadap gerakan, sehingga diperlukan rata-rata dalam pengambilan fitur di aplikasi ini.

6.2 Saran

Berikut saran-saran untuk pengembangan dan perbaikan sistem di masa yang akan datang. Di antaranya adalah sebagai berikut:

1. Perlu adanya penelitian lebih lanjut tentang proses kalibrasi dari jari tangan setiap pengguna. Sehingga dapat meningkatkan

nilai akurasi aplikasi ini. Hal ini harus dilakukan karena pada saat *testing* terdapat perbedaan bentuk, panjang, lebar masing-masing fitur setiap jari pengguna.

2. Memperbanyak model *neural network* dengan cara melakukan *training* dari berbagai jari pengguna.

DAFTAR PUSTAKA

[1] A. W. Yanuardi, P. Samudra and J. A. Purnama P., "Indonesian Sign Language Computer Application for The Deaf," *IEEE*, 2010.

[2] N. Ulfah, "5.000 Bayi Indonesia Lahir Tuli Setiap Tahun," Detik Health, 9 Januari 2010. [Online]. Available: http://health.detik.com/read/2010/01/09/155558/1274969/763/. [Accessed 26 Desember 2014].

[3] L. Motion, "Leap Motion," Leap Motion Inc., 2012. [Online]. Available: https://developer.leapmotion.com/documentation/csharp/index.html. [Accessed 22 Desember 2014].

[4] S. Aliyu, M. Mohandes and M. Deriche, "Arabic Sign Language Recognition using the Leap Motion Controller," *IEEE*, 2014.

[5] O. Mencer. [Online]. Available: http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html. [Accessed 6 Juni 2015].

[6] Aberdeen's Robert Gordon University, "www.rgu.ac.uk," [Online]. Available: https://www4.rgu.ac.uk/files/chapter3%20-%20bp.pdf. [Accessed 2 Maret 2015].

[7] S. Wang, S. Yin and M. Jiang, "Hybrid Neural Network Based On GA-BP for Personal Credit Scoring," *Fourth International Conference on Natural Computation*, 2008.

[8] E. Rakun, M. Andriarni, I. W. Wiprayoga, K. Danniswara and A. Tjandra, "Combining Depth Image and Skeleton Data from Kinect for Recognizing Words in the Sign System for Indonesian Language (SIBI)," *IEEE*, 2013.

BAB VI
KESIMPULAN DAN SARAN

Pada bab ini akan diberikan kesimpulan yang diambil selama pengerjaan tugas akhir serta saran-saran tentang pengembangan yang dapat dilakukan terhadap tugas akhir ini di masa yang akan datang.

6.1 Kesimpulan

Dari proses pengerjaan selama perancangan, implementasi, dan proses pengujian aplikasi yang dilakukan, dapat diambil kesimpulan sebagai berikut.

1. Dari sisi akurasi, penerapan algoritma GABP pada aplikasi ini lebih akurat daripada algoritma BP. Algoritma GABP dapat meningkatkan akurasi *neural network* sekitar 7-13%.
2. Dari sisi *runtime*, penerapan algoritma GABP pada aplikasi ini lebih cepat daripada algoritma BP.
3. Aplikasi yang dibangun pada tugas akhir ini dapat menerjemahkan bahasa isyarat huruf dengan akurasi di atas 80% jika pada saat *testing* menggunakan data *training* tangan yang sama. Namun, tangan yang ada pada data *training* berbeda pada saat *testing*, akurasi aplikasi ini menjadi sekitar 65-75%.
4. Leap Motion merupakan alat yang sangat sensitif terhadap gerakan, sehingga diperlukan rata-rata dalam pengambilan fitur di aplikasi ini.

6.2 Saran

Berikut saran-saran untuk pengembangan dan perbaikan sistem di masa yang akan datang. Di antaranya adalah sebagai berikut:

1. Perlu adanya penelitian lebih lanjut tentang proses kalibrasi dari jari tangan setiap pengguna. Sehingga dapat meningkatkan

nilai akurasi aplikasi ini. Hal ini harus dilakukan karena pada saat *testing* terdapat perbedaan bentuk, panjang, lebar masing-masing fitur setiap jari pengguna.

2. Memperbanyak model *neural network* dengan cara melakukan *training* dari berbagai jari pengguna.

LAMPIRAN A KODE SUMBER

```
class BackPropagation
{
private float learningRate = 0.03f;
private FeedForward feedForward;
private NeuralNetwork lastStableNetwork;
public NeuralNetwork Network{
 set{
  feedForward.Network = value;
 }
 get{
  return feedForward.Network;
 }
}
public ListDataSet DataSetList
{
 get{
  return feedForward.DataSetList;
 }
}
private ClassificationClass classificationClass;

public void Initialise(NeuralNetwork nn, ListDataSet
dsl, ClassificationClass cc){
 feedForward = new FeedForward();
 feedForward.Initialise(nn, dsl);
 classificationClass = cc;
}

public float Run(int maxIter)
{
 float lastAvgError = float.MaxValue;
 float avgError =0;
 for (int it = 0; it < maxIter; it++){
 avgError = 0;
 for (int k = 0; k < DataSetList.Count; k++){
  Network.InitaliseInput();
  feedForward.Run(k);
  float[] errorOutput = GetErrorOutput(k);
```

```
  float totalError = 0;
  for (int i = 0; i < errorOutput.Length; i++){
   totalError += Math.Abs(errorOutput[i]);
  }
  avgError += totalError;
 }
 avgError /= DataSetList.Count;
 }
 return avgError;
}

private float[] GetErrorOutput(int index){
  float[] outputError = new
float[classificationClass.TargetCount];
  for (int i = 0; i <
classificationClass.TargetCount; i++){
    outputError[i] =
classificationClass.GetTarget(DataSetList[index].Clas
sName)[i] - Network.OutputLayer[i].Input;
  }
  UpdateWeightHidden(outputError, index);
  return outputError;
}

private void UpdateWeightInput(float[] hiddenError, int index){
for (int i = 0; i < Network.InputLayer.Count; i++){
for (int j = 0; j < Network.HiddenLayer.Count; j++){
float newWeight = Network.InputLayer[i].GetWeight(j) + (learningRate
* hiddenError[j] * Network.InputLayer[i].Input);
Network.InputLayer[i].SetWeight(j, newWeight);
 }
 }
}
private void UpdateWeightHidden(float[] outputError,
int index){
for (int i = 0; i < Network.HiddenLayer.Count; i++){
 for (int j = 0; j < Network.OutputLayer.Count; j++){
 float newWeight =
Network.HiddenLayer[i].GetWeight(j) +
```

```
(this.learningRate * outputError[j] *
Network.HiddenLayer[i].Input);
Network.HiddenLayer[i].SetWeight(j, newWeight);
  }
 }
GetErrorHidden(outputError, index);
}

private void GetErrorHidden(float[] outputError, int
index){
float[] hiddenError = new
float[Network.HiddenLayer.Count];
for (int i = 0; i < Network.HiddenLayer.Count; i++){
 float linear = 0;
 for (int j = 0; j < Network.OutputLayer.Count; j++){
 linear += outputError[j] *
Network.HiddenLayer[i].GetWeight(j);
 }
 hiddenError[i] = Network.HiddenLayer[i].Input * (1 -
Network.HiddenLayer[i].Input) * linear;
 }
 UpdateWeightInput(hiddenError, index);
}
```

Kode Sumber 7. 1 Kelas Backpropagation.cs

```
class Chromosom
{
 private int[] bit;
 public int[] Bit{
 set{
       bit = value;
 }
 get{
  return bit;
 }
}
```

```
 public int this[int index]{
 set{
  bit[index] = value;
 }
 get{
  return bit[index];
 }
 }

 public int Length{
 get{
  return bit.Length;
 }
 }

 private float fitnessValue = 0;
 public float FitnessValue{
  set{
  fitnessValue = value;
 }
 get{
  return fitnessValue;
 }
 }

 public Chromosom(int[] newBit){
 bit = newBit;
 }

 public void Print(){
  for (int i = 0; i < bit.Length; i++){
   Console.Write(bit[i] + " ");
  }
 }
}
```

Kode Sumber 7. 2 Kelas Chromosom.cs

```
class ClassificationClass
{
private List<string> classList = new List<string>();
private int[] actualClass;
public int TargetCount
{
	get
	{
		int factor = 1;
		while (Math.Pow(2, factor) <
classList.Count)
			factor++;
		return factor;
	}
}

public void Add(string className)
{
	int index = GetIndex(className);
	if (index == -1)
	{
		classList.Add(className);
	}
}

public void Clear()
{
	classList.Clear();
}

public int GetIndex(string className)
{
	return classList.IndexOf(className);
}
public int[] GetTarget(int index)
{
	int[] target = new int[TargetCount];
	int bitCount = target.Length - 1;
```

```
        while (index > 0)
        {
                target[bitCount--] = (index % 2);
                index = index / 2;
        }

        return target;
}
public int[] GetTarget(string className)
{
        return GetTarget(GetIndex(className));
}
}
```

Kode Sumber 7. 3 Kelas ClassificationClass.cs

```
class DataSet
    {
        public string ClassName { set; get; }
        private List<float> _attribute;
        public float this[int index]
        {
            set
            {
                _attribute[index] = value;
            }
            get
            {
                return _attribute[index];
            }
        }
        public int AttributeCount
        {
            get
            {
                return _attribute.Count;
            }
        }
```

```
        public DataSet()
        {
            _attribute = new List<float>();
        }
        public DataSet(int attributeCount)
        {
            _attribute = new List<float>();
            for (int i = 0; i < attributeCount; i++)
                _attribute.Add(0);
        }
        public DataSet(DataSet ds)
        {
            _attribute = new List<float>();

            for (int i = 0; i < ds.AttributeCount;
i++)
            {
                _attribute.Add(ds[i]);
            }

            this.ClassName = ds.ClassName;
        }
        public void RemoveBit(int index)
        {
            _attribute.RemoveAt(index);
        }
    }
```

Kode Sumber 7. 4 Kelas DataSet.cs

```
class FeedForward
{
 private NeuralNetwork neuralNetwork;
 public NeuralNetwork Network
 {
 set{
 neuralNetwork = value;
 }
 get{
 return neuralNetwork;
 }
 }

 private ListDataSet dataSetList;
 public ListDataSet DataSetList
 {
 get{
  return dataSetList;
 }
 }

 public void Initialise(NeuralNetwork nn, ListDataSet
dsl){
       neuralNetwork = nn;
       dataSetList = dsl;
 }

 public void Run(int index){
       neuralNetwork.InitaliseInput();
       DoInputLayer(index);
 }

public void Initialise(NeuralNetwork nn, ListDataSet
dsl){
       neuralNetwork = nn;
       dataSetList = dsl;
 }

 public void Run(int index){
       neuralNetwork.InitaliseInput();
```

```
        DoInputLayer(index);
 }

private void DoInputLayer(int index){
 for (int i = 0; i <
dataSetList[index].AttributeCount; i++){
  neuralNetwork.InputLayer[i].Input =
dataSetList[index][i];
 }
 DoHiddenLayer();
}

private void DoHiddenLayer()
{
 for (int i = 0; i < neuralNetwork.InputLayer.Count;
i++){
 for (int j = 0; j < neuralNetwork.HiddenLayer.Count;
j++){
  neuralNetwork.HiddenLayer[j].Input +=
neuralNetwork.InputLayer[i].GetOutput(j);
  }
 }
 for (int j = 0; j < neuralNetwork.HiddenLayer.Count;
j++){
 neuralNetwork.HiddenLayer[j].Input =
Sigmoid(neuralNetwork.HiddenLayer[j].Input);
  }
  DoOutputLayer();
}

private void DoOutputLayer(){
for (int i = 0; i < neuralNetwork.HiddenLayer.Count;
i++){
 for (int j = 0; j < neuralNetwork.OutputLayer.Count;
j++)
 {
 neuralNetwork.OutputLayer[j].Input +=
neuralNetwork.HiddenLayer[i].Input *
neuralNetwork.HiddenLayer[i].GetWeight(j);
  }
```

```
 }
}

private float Sigmoid(float val){
 return 1 / (1.0f + (float)Math.Exp(-val));
}
```

Kode Sumber 7. 5 Kelas FeedForward.cs

```
class GeneticAlgorithm
{
 const int individualCount = 3;
 private Random random = new Random();
 private ClassificationClass classificationClass;
 private BackPropagation backPropagation;
 private NeuralNetwork network;
 private ListDataSet listDataSet;
 private List<Chromosom> chromosoms;

 public void Initialize(ListDataSet lds,
ClassificationClass cc){
  listDataSet = lds;
  classificationClass = cc;
 }

 public Chromosom Run(){
 ChromosomInit();
 int iteration = 3;
 for (int i = 0; i < iteration; i++){
  DoBackPropagation();
  DoSelection();
  DoCrossOver();
 int chanceMutation = GetRandom();
 if (chanceMutation < GetRandom()){
  DoMutation();
  }
 }
```

```
 int index = 0;
 for (int i = 1; i < chromosoms.Count; i++){
 if (chromosoms[i].FitnessValue >
chromosoms[index].FitnessValue){
  index = i;
  }
 }

 Chromosom fittestChromosom = chromosoms[index];
 return fittestChromosom;
 }

 private void ChromosomInit()
 {
  chromosoms = new List<Chromosom>(individualCount);
  for (int i = 0; i < individualCount; i++){
    chromosoms.Add(new Chromosom(GetRandomBinary()));
  }
 }

 private void DoSelection(){
  chromosoms.Sort((val1, val2) =>
val1.FitnessValue.CompareTo(val2.FitnessValue));
 }

 private void DoCrossOver(){
 for (int i = 0; i < chromosoms.Count; i += 2){
 int nextIndex = i + 1;
 if (nextIndex >= chromosoms.Count)
 nextIndex = i - 1;
 int[] bit1 = chromosoms[i].Bit;
 int[] bit2 = chromosoms[nextIndex].Bit;
 for (int j = bit1.Length / 2; j < bit1.Length; j++){
  int t = bit1[j];
  bit1[j] = bit2[j];
  bit2[j] = t;
  }
 chromosoms[i].Bit = bit1;
 chromosoms[nextIndex].Bit = bit2;
 }
```

```
 }

 private void DoMutation()
 {
 for (int i = 0; i < chromosoms.Count; i++){
  int chanceChromoMut = GetRandom();
  if (chanceChromoMut <= GetRandom()){
   int bitIndex = GetRandom(chromosoms[i].Length);
   if (chromosoms[i][bitIndex] == 0)
chromosoms[i][bitIndex] = 1;
   else chromosoms[i][bitIndex] = 0;
  }
 }
 }

 private void DoBackPropagation(){
  for (int i = 0; i < chromosoms.Count; i++){
  ListDataSet lds = new ListDataSet(listDataSet);
  for (int j = 0; j < lds.Count; j++){
  int popCount = 0;
  for (int k = 0; k < chromosoms[i].Length; k++){
   if (chromosoms[i][k] == 0){
   lds[j].RemoveBit(k - popCount);
   popCount++;
    }
   }
  }
  NeuralNetwork nn = new NeuralNetwork();
  nn.InitialiseNetwork(lds[0].AttributeCount,
lds[0].AttributeCount / 2,
classificationClass.TargetCount);
  nn.InitialiseWeight();

  BackPropagation bp = new BackPropagation();
  bp.Initialise(nn, lds, classificationClass);
  bp.Run(5000);

  FeedForward ff = new FeedForward();
  ff.Initialise(nn, lds);
```

```
  int totalCorrect = 0;
  for (int j = 0; j < lds.Count; j++){
    ff.Run(j);
       bool correct = true;
       int[] targetClass =
classificationClass.GetTarget(lds[j].ClassName);
       for (int k = 0; k <
ff.GetActualClass().Length; k++){
         if (targetClass[k] !=
ff.GetActualClass()[k])
          correct = false;
         }
        if (correct)totalCorrect++;
        }
       chromosoms[i].FitnessValue = totalCorrect /
(float)lds.Count;
       }
 }
}
```

Kode Sumber 7. 6 Kelas GeneticAlgorithm

```
class NeuralNetwork
{
private Random random = new Random();
public int Seed{
          set{
           random = new Random(value);
          }
}

public float treshold = 0.5f;
public List<Node> InputLayer { set; get; }
public List<Node> HiddenLayer { set; get; }
public List<Node> OutputLayer { set; get; }

public NeuralNetwork(){
}
```

```
public NeuralNetwork(NeuralNetwork newNN){
        newNN.InitaliseInput();
        InitialiseNetwork(newNN.InputLayer.Count,
newNN.HiddenLayer.Count, newNN.OutputLayer.Count);
        for (int i = 0; i < newNN.InputLayer.Count; i++){
                InputLayer[i] = newNN.InputLayer[i];
        }

        for (int i = 0; i < newNN.HiddenLayer.Count; i++){
                HiddenLayer[i] = newNN.HiddenLayer[i];
        }
}

public void InitialiseNetwork(int inputLayerCount, int
hiddenLayerCount, int outputLayerCount){
        InputLayer = new List<Node>();
        HiddenLayer = new List<Node>();
        OutputLayer = new List<Node>();

        for (int i = 0; i < inputLayerCount; i++){
                InputLayer.Add(new Node(hiddenLayerCount));
        }

        for (int i = 0; i < hiddenLayerCount; i++){
                HiddenLayer.Add(new Node(outputLayerCount));
        }

        for (int i = 0; i < outputLayerCount; i++){
                OutputLayer.Add(new Node(1));
                OutputLayer[i].SetWeight(0, 1);
        }
}

public void InitaliseInput()
{
        for (int i = 0; i < InputLayer.Count; i++){
                InputLayer[i].Input = 0;
        }

        for (int i = 0; i < HiddenLayer.Count; i++){
                HiddenLayer[i].Input = 0;
```

```
        }

        for (int i = 0; i < OutputLayer.Count; i++){
                OutputLayer[i].Input = 0;
        }
}

public void InitialiseWeight(){
        for (int i = 0; i < InputLayer.Count; i++){
                for (int j = 0; j < HiddenLayer.Count; j++){
                        InputLayer[i].SetWeight(j, GetRandom() /
100.0f);
                }
        }

        for (int i = 0; i < HiddenLayer.Count; i++){
                for (int j = 0; j < OutputLayer.Count; j++){
                        HiddenLayer[i].SetWeight(j, GetRandom() /
100.0f);
                }
        }
}

int GetRandom(){
        return random.Next(0, 30);
}
}
```

Kode Sumber 7. 7 Kelas NeuralNetwork.cs

LAMPIRAN A KODE SUMBER

```
class BackPropagation
{
private float learningRate = 0.03f;
private FeedForward feedForward;
private NeuralNetwork lastStableNetwork;
public NeuralNetwork Network{
 set{
  feedForward.Network = value;
 }
 get{
  return feedForward.Network;
 }
}
public ListDataSet DataSetList
{
 get{
  return feedForward.DataSetList;
 }
}
private ClassificationClass classificationClass;

public void Initialise(NeuralNetwork nn, ListDataSet
dsl, ClassificationClass cc){
 feedForward = new FeedForward();
 feedForward.Initialise(nn, dsl);
 classificationClass = cc;
}

public float Run(int maxIter)
{
 float lastAvgError = float.MaxValue;
 float avgError =0;
 for (int it = 0; it < maxIter; it++){
 avgError = 0;
 for (int k = 0; k < DataSetList.Count; k++){
  Network.InitaliseInput();
  feedForward.Run(k);
  float[] errorOutput = GetErrorOutput(k);
```

```
  float totalError = 0;
  for (int i = 0; i < errorOutput.Length; i++){
   totalError += Math.Abs(errorOutput[i]);
  }
  avgError += totalError;
 }
 avgError /= DataSetList.Count;
 }
 return avgError;
}

private float[] GetErrorOutput(int index){
  float[] outputError = new
float[classificationClass.TargetCount];
  for (int i = 0; i <
classificationClass.TargetCount; i++){
    outputError[i] =
classificationClass.GetTarget(DataSetList[index].Clas
sName)[i] - Network.OutputLayer[i].Input;
  }
  UpdateWeightHidden(outputError, index);
  return outputError;
}

private void UpdateWeightInput(float[] hiddenError, int index){
for (int i = 0; i < Network.InputLayer.Count; i++){
for (int j = 0; j < Network.HiddenLayer.Count; j++){
float newWeight = Network.InputLayer[i].GetWeight(j) + (learningRate
* hiddenError[j] * Network.InputLayer[i].Input);
Network.InputLayer[i].SetWeight(j, newWeight);
 }
 }
}
private void UpdateWeightHidden(float[] outputError,
int index){
for (int i = 0; i < Network.HiddenLayer.Count; i++){
 for (int j = 0; j < Network.OutputLayer.Count; j++){
 float newWeight =
Network.HiddenLayer[i].GetWeight(j) +
```

```
(this.learningRate * outputError[j] *
Network.HiddenLayer[i].Input);
Network.HiddenLayer[i].SetWeight(j, newWeight);
  }
 }
GetErrorHidden(outputError, index);
}

private void GetErrorHidden(float[] outputError, int
index){
float[] hiddenError = new
float[Network.HiddenLayer.Count];
for (int i = 0; i < Network.HiddenLayer.Count; i++){
 float linear = 0;
 for (int j = 0; j < Network.OutputLayer.Count; j++){
 linear += outputError[j] *
Network.HiddenLayer[i].GetWeight(j);
 }
 hiddenError[i] = Network.HiddenLayer[i].Input * (1 -
Network.HiddenLayer[i].Input) * linear;
 }
 UpdateWeightInput(hiddenError, index);
}
```

Kode Sumber 7. 1 Kelas Backpropagation.cs

```
class Chromosom
{
 private int[] bit;
 public int[] Bit{
 set{
       bit = value;
 }
 get{
  return bit;
 }
}
```

```
 public int this[int index]{
 set{
  bit[index] = value;
 }
 get{
  return bit[index];
 }
 }

 public int Length{
 get{
  return bit.Length;
 }
 }

 private float fitnessValue = 0;
 public float FitnessValue{
  set{
  fitnessValue = value;
 }
 get{
  return fitnessValue;
 }
 }

 public Chromosom(int[] newBit){
 bit = newBit;
 }

 public void Print(){
  for (int i = 0; i < bit.Length; i++){
   Console.Write(bit[i] + " ");
  }
 }
}
```

Kode Sumber 7. 2 Kelas Chromosom.cs

```
class ClassificationClass
{
private List<string> classList = new List<string>();
private int[] actualClass;
public int TargetCount
{
	get
	{
		int factor = 1;
		while (Math.Pow(2, factor) <
classList.Count)
			factor++;
		return factor;
	}
}

public void Add(string className)
{
	int index = GetIndex(className);
	if (index == -1)
	{
		classList.Add(className);
	}
}

public void Clear()
{
	classList.Clear();
}

public int GetIndex(string className)
{
	return classList.IndexOf(className);
}
public int[] GetTarget(int index)
{
	int[] target = new int[TargetCount];
	int bitCount = target.Length - 1;
```

```
        while (index > 0)
        {
                target[bitCount--] = (index % 2);
                index = index / 2;
        }

        return target;
}
public int[] GetTarget(string className)
{
        return GetTarget(GetIndex(className));
}
}
```

Kode Sumber 7. 3 Kelas ClassificationClass.cs

```
class DataSet
    {
        public string ClassName { set; get; }
        private List<float> _attribute;
        public float this[int index]
        {
            set
            {
                _attribute[index] = value;
            }
            get
            {
                return _attribute[index];
            }
        }
        public int AttributeCount
        {
            get
            {
                return _attribute.Count;
            }
        }
```

```
        public DataSet()
        {
            _attribute = new List<float>();
        }
        public DataSet(int attributeCount)
        {
            _attribute = new List<float>();
            for (int i = 0; i < attributeCount; i++)
                _attribute.Add(0);
        }
        public DataSet(DataSet ds)
        {
            _attribute = new List<float>();

            for (int i = 0; i < ds.AttributeCount;
i++)
            {
                _attribute.Add(ds[i]);
            }

            this.ClassName = ds.ClassName;
        }
        public void RemoveBit(int index)
        {
            _attribute.RemoveAt(index);
        }
    }
```

Kode Sumber 7. 4 Kelas DataSet.cs

```
class FeedForward
{
 private NeuralNetwork neuralNetwork;
 public NeuralNetwork Network
 {
 set{
 neuralNetwork = value;
 }
 get{
 return neuralNetwork;
 }
 }

 private ListDataSet dataSetList;
 public ListDataSet DataSetList
 {
 get{
  return dataSetList;
 }
 }

 public void Initialise(NeuralNetwork nn, ListDataSet
dsl){
       neuralNetwork = nn;
       dataSetList = dsl;
 }

 public void Run(int index){
       neuralNetwork.InitaliseInput();
       DoInputLayer(index);
 }

public void Initialise(NeuralNetwork nn, ListDataSet
dsl){
       neuralNetwork = nn;
       dataSetList = dsl;
 }

 public void Run(int index){
       neuralNetwork.InitaliseInput();
```

```
        DoInputLayer(index);
 }

private void DoInputLayer(int index){
 for (int i = 0; i <
dataSetList[index].AttributeCount; i++){
  neuralNetwork.InputLayer[i].Input =
dataSetList[index][i];
 }
 DoHiddenLayer();
}

private void DoHiddenLayer()
{
 for (int i = 0; i < neuralNetwork.InputLayer.Count;
i++){
 for (int j = 0; j < neuralNetwork.HiddenLayer.Count;
j++){
  neuralNetwork.HiddenLayer[j].Input +=
neuralNetwork.InputLayer[i].GetOutput(j);
  }
 }
 for (int j = 0; j < neuralNetwork.HiddenLayer.Count;
j++){
 neuralNetwork.HiddenLayer[j].Input =
Sigmoid(neuralNetwork.HiddenLayer[j].Input);
  }
  DoOutputLayer();
}

private void DoOutputLayer(){
for (int i = 0; i < neuralNetwork.HiddenLayer.Count;
i++){
 for (int j = 0; j < neuralNetwork.OutputLayer.Count;
j++)
 {
 neuralNetwork.OutputLayer[j].Input +=
neuralNetwork.HiddenLayer[i].Input *
neuralNetwork.HiddenLayer[i].GetWeight(j);
  }
```

```
 }
}

private float Sigmoid(float val){
 return 1 / (1.0f + (float)Math.Exp(-val));
}
```

Kode Sumber 7. 5 Kelas FeedForward.cs

```
class GeneticAlgorithm
{
 const int individualCount = 3;
 private Random random = new Random();
 private ClassificationClass classificationClass;
 private BackPropagation backPropagation;
 private NeuralNetwork network;
 private ListDataSet listDataSet;
 private List<Chromosom> chromosoms;

 public void Initialize(ListDataSet lds,
ClassificationClass cc){
  listDataSet = lds;
  classificationClass = cc;
 }

 public Chromosom Run(){
 ChromosomInit();
 int iteration = 3;
 for (int i = 0; i < iteration; i++){
  DoBackPropagation();
  DoSelection();
  DoCrossOver();
 int chanceMutation = GetRandom();
 if (chanceMutation < GetRandom()){
  DoMutation();
  }
 }
```

```
 int index = 0;
 for (int i = 1; i < chromosoms.Count; i++){
 if (chromosoms[i].FitnessValue >
chromosoms[index].FitnessValue){
  index = i;
  }
 }

 Chromosom fittestChromosom = chromosoms[index];
 return fittestChromosom;
 }

 private void ChromosomInit()
 {
  chromosoms = new List<Chromosom>(individualCount);
  for (int i = 0; i < individualCount; i++){
    chromosoms.Add(new Chromosom(GetRandomBinary()));
  }
 }

 private void DoSelection(){
  chromosoms.Sort((val1, val2) =>
val1.FitnessValue.CompareTo(val2.FitnessValue));
 }

 private void DoCrossOver(){
 for (int i = 0; i < chromosoms.Count; i += 2){
 int nextIndex = i + 1;
 if (nextIndex >= chromosoms.Count)
 nextIndex = i - 1;
 int[] bit1 = chromosoms[i].Bit;
 int[] bit2 = chromosoms[nextIndex].Bit;
 for (int j = bit1.Length / 2; j < bit1.Length; j++){
  int t = bit1[j];
  bit1[j] = bit2[j];
  bit2[j] = t;
  }
 chromosoms[i].Bit = bit1;
 chromosoms[nextIndex].Bit = bit2;
 }
```

```
 }

 private void DoMutation()
 {
 for (int i = 0; i < chromosoms.Count; i++){
  int chanceChromoMut = GetRandom();
  if (chanceChromoMut <= GetRandom()){
   int bitIndex = GetRandom(chromosoms[i].Length);
   if (chromosoms[i][bitIndex] == 0)
chromosoms[i][bitIndex] = 1;
   else chromosoms[i][bitIndex] = 0;
  }
 }
 }

 private void DoBackPropagation(){
  for (int i = 0; i < chromosoms.Count; i++){
  ListDataSet lds = new ListDataSet(listDataSet);
  for (int j = 0; j < lds.Count; j++){
  int popCount = 0;
  for (int k = 0; k < chromosoms[i].Length; k++){
   if (chromosoms[i][k] == 0){
   lds[j].RemoveBit(k - popCount);
   popCount++;
    }
   }
  }
  NeuralNetwork nn = new NeuralNetwork();
  nn.InitialiseNetwork(lds[0].AttributeCount,
lds[0].AttributeCount / 2,
classificationClass.TargetCount);
  nn.InitialiseWeight();

  BackPropagation bp = new BackPropagation();
  bp.Initialise(nn, lds, classificationClass);
  bp.Run(5000);

  FeedForward ff = new FeedForward();
  ff.Initialise(nn, lds);
```

```
  int totalCorrect = 0;
  for (int j = 0; j < lds.Count; j++){
    ff.Run(j);
       bool correct = true;
       int[] targetClass =
classificationClass.GetTarget(lds[j].ClassName);
       for (int k = 0; k <
ff.GetActualClass().Length; k++){
         if (targetClass[k] !=
ff.GetActualClass()[k])
          correct = false;
         }
        if (correct)totalCorrect++;
        }
       chromosoms[i].FitnessValue = totalCorrect /
(float)lds.Count;
       }
 }
}
```

Kode Sumber 7. 6 Kelas GeneticAlgorithm

```
class NeuralNetwork
{
private Random random = new Random();
public int Seed{
          set{
           random = new Random(value);
          }
}

public float treshold = 0.5f;
public List<Node> InputLayer { set; get; }
public List<Node> HiddenLayer { set; get; }
public List<Node> OutputLayer { set; get; }

public NeuralNetwork(){
}
```

```
public NeuralNetwork(NeuralNetwork newNN){
        newNN.InitaliseInput();
        InitialiseNetwork(newNN.InputLayer.Count,
newNN.HiddenLayer.Count, newNN.OutputLayer.Count);
        for (int i = 0; i < newNN.InputLayer.Count; i++){
                InputLayer[i] = newNN.InputLayer[i];
        }

        for (int i = 0; i < newNN.HiddenLayer.Count; i++){
                HiddenLayer[i] = newNN.HiddenLayer[i];
        }
}

public void InitialiseNetwork(int inputLayerCount, int
hiddenLayerCount, int outputLayerCount){
        InputLayer = new List<Node>();
        HiddenLayer = new List<Node>();
        OutputLayer = new List<Node>();

        for (int i = 0; i < inputLayerCount; i++){
                InputLayer.Add(new Node(hiddenLayerCount));
        }

        for (int i = 0; i < hiddenLayerCount; i++){
                HiddenLayer.Add(new Node(outputLayerCount));
        }

        for (int i = 0; i < outputLayerCount; i++){
                OutputLayer.Add(new Node(1));
                OutputLayer[i].SetWeight(0, 1);
        }
}

public void InitaliseInput()
{
        for (int i = 0; i < InputLayer.Count; i++){
                InputLayer[i].Input = 0;
        }

        for (int i = 0; i < HiddenLayer.Count; i++){
                HiddenLayer[i].Input = 0;
```

```
        }

        for (int i = 0; i < OutputLayer.Count; i++){
                OutputLayer[i].Input = 0;
        }
}

public void InitialiseWeight(){
        for (int i = 0; i < InputLayer.Count; i++){
                for (int j = 0; j < HiddenLayer.Count; j++){
                        InputLayer[i].SetWeight(j, GetRandom() /
100.0f);
                }
        }

        for (int i = 0; i < HiddenLayer.Count; i++){
                for (int j = 0; j < OutputLayer.Count; j++){
                        HiddenLayer[i].SetWeight(j, GetRandom() /
100.0f);
                }
        }
}

int GetRandom(){
        return random.Next(0, 30);
}
}
```

Kode Sumber 7. 7 Kelas NeuralNetwork.cs

RANCANG BANGUN MODUL PENGENALAN BAHASA ISYARAT MENGGUNAKAN TEKNOLOGI LEAP MOTION DAN METODE BACKPROPAGATION-GENETIC ALGORITHM NEURAL NETWORK

Nama Mahasiswa : Risal Andika Tridisaputra
NRP : 5111100133
Jurusan : Teknik Informatika FTIf-ITS
Dosen Pembimbing I : Wijayanti Nurul Khotimah, S.Kom, M.Sc.
Dosen Pembimbing II : Ridho Rahman Hariadi, S.Kom., M.Sc.

ABSTRAK

Bahasa isyarat adalah hal yang penting dalam komunikasi bagi orang yang menderita gangguan pendengaran. Kecepatan menguasai bahasa dan kemampuan mereka berinteraksi sangat dibutuhkan. Mereka membutuhkan bahan pembelajaran yang tidak hanya berisi tentang komponen aural saja, namun juga secara visual karena lebih nyata.

Di sisi lain, teknologi berkembang pesat di segala aspek kehidupan. Berbagai macam terobosan teknologi baru telah diciptakan oleh manusia, salah satunya perangkat Leap Motion yang diciptakan pada sekitar tahun 2013. Dengan menggunakan Leap Motion, manusia dapat melakukan interaksi dengan komputer tanpa sentuhan sama sekali.

Oleh karena itu, penulis memiliki ide yang digunakan dalam tugas akhir ini yaitu akan dibangun sebuah aplikasi pengenalan huruf bahasa isyarat Indonesia sesuai dengan SIBI (Sistem Isyarat Bahasa Indonesia). Tujuannya diharapkan aplikasi ini dapat membantu pengguna belajar tentang bahasa isyarat. Aplikasi ini dapat menerjemahkan bahasa isyarat huruf dengan akurasi di atas 80% jika tangan yang digunakan sama ketika testing. Jika orang yang melakukan testing berbeda, akurasi aplikasi ini sekitar 65-75%.

Kata kunci: Leap Motion, bahasa isyarat, SIBI.

DESIGN AND IMPLEMENTATION OF INDONESIAN SIGN LANGUAGE WITH LEAP MOTION CONTROLLER AND BACKPROPAGATION-GENETIC ALGORITHM NEURAL NETWORK METHOD

Name : **Risal Andika Tridisaputra**
NRP : **5111100133**
Major : **Informatics Department, FTIf-ITS**
Advisor I : **Wijayanti Nurul Khotimah, S.Kom, M.Sc.**
Advisor II : **Ridho Rahman Hariadi, S.Kom., M.Sc.**

ABSTRACT

Sign language is an important thing in the communication of deaf people. The speed of learning sign language and their skills to interact each other is very necessary. They need a media and the learning materials is not only about aural matter, but also visual.

On the other hand, technology growth very quickly. The new Many technology are invented, one of them is Leap Motion which invented in 2013. With Leap Motion Controller, human can interact with computer touchless.

Because of that,the writer has an idea on this final project, it is an application that can be used to learn about Indonesian sign language based on SIBI (Sistem Isyarat Bahasa Indonesia). This application is developed by Microsoft Visual Studio with Leap Motion SDK. The goal of this application is to help people know and learn about Indonesian sign language.The accuracy of this application is more than 80% when the test using the same hand. If it used by another person, the accuracy of this application become 65-75%.

Keywords: : Leap Motion, Sign Language, SIBI

KATA PENGANTAR

Dengan menyebut nama Allah SWT yang Maha Pengasih lagi Maha Panyayang, kami panjatkan puja dan puji syukur atas kehadirat-Nya, yang telah melimpahkan rahmat, hidayah, dan inayah-Nya kepada kami, sehingga penulis dapat menyelesaikan tugas akhir ini dengan baik.

Penulis ingin menyampaikan rasa hormat dan terima kasih yang setinggi-tingginya kepada pihak-pihak yang telah membantu penulis dalam penyelesaian tugas akhir ini, terutama kepada:

1. Bapak Didi Wahyu Rahmadi dan Ibu Nurqolisa Diah Permata, orang tua penulis, yang selalu memberikan dukungan dan semangat baik dalam bantuan finansial maupun dorongan positif agar penulis mampu untuk menyelesaikan tugas akhir dengan benar dan tepat waktu, serta memberikan doa yang terus dikirimkan agar penyelesaian tugas akhir berjalan dengan lancar.
2. Kedua kakak-kakak saya Ryan Ditya Permadi dan Riska Amanda Dwinindita, yang selalu memberikan semangat dan memberikan doa agar penulis mampu menyelesaikan Tugas Akhirnya.
3. Ibu Henning Titi Ciptaningtyas, S.Kom., M.Kom. sebagai dosen wali penulis yang turut memberikan saran dan menuntun penulis dalam menentukan mata kuliah yang akan diambil pada pembelajaran di Jurusan Teknik Informatika ITS ini.
4. Ibu Wijayanti Nurul Khotimah, S.Kom., M.Sc. dan Bapak Ridho Rahman Hariadi, S.Kom., M.Sc. yang telah bersedia untuk menjadi dosen pembimbing tugas akhir sehingga penulis dapat mengerjakan tugas akhir dengan arahan dan bimbingan yang baik dan jelas.
5. Teman-teman Mahasiswa Teknis Informatika 2011 yang telah berjuang bersama-sama selama menempuh pendidikan di Jurusan ini.

6. Teman-teman administrator Lab Komputasi Berbasis Jaringan yaitu Ade, Helmy, Nisa, Deasy, Billa, dan adik-adik yang selalu memberikan inspirasi tersendiri kepada penulis.
7. Teman-teman kontrakan, Rahman, Tommy, Ruslan, Faris, Punggi, Dapik, Andrie, Toto di Wisma Permai yang selalu memberi hiburan kepada penulis.
8. Teman-teman panitia pengurus SCHEMATICS 2013 khususnya Besta, Petrus, Jordy, Baskara, Indra, Ali, Ihe, Risma, Harum, Raras, Uthe dan teman-teman lain yang sudah bersusah payah mengadakan acara untuk mengharumkan jurusan kita.
9. Teman-teman seperjuangan angkatan 2011 Teknik Informatika ITS.
10. Serta pihak-pihak lain yang turut membantu penulis baik secara langsung maupun tidak, yang namanya tidak penulis sebutkan disini.

Penulis telah berusaha sebaik-baiknya dalam menyusun tugas akhir ini, mohon maaf apabila ada kesalahan dan kata-kata yang dapat menyinggung perasaan. Penulis berharap tugas akhir ini dapat menjadi media pembelajaran bahasa isyarat Indonesia.

Surabaya, Juni 2015

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR KODE SUMBER

BIODATA PENULIS

Penulis, Risal Andika Tridisaputra lahir di Jember, Jawa Timur, pada tanggal 9 Juni 1993. Penulis adalah anak ke-3 dari 3 bersaudara dan dibesarkan di Kota Surabaya. Penulis menempuh pendidikan formal di SDN AL-Falah Surabaya (1999-2005), SMPN 32 Surabaya (2005-2008), dan SMAN 2 Surabaya (2008-2011). Pada tahun 2011, penulis menempuh pendidikan S1 jurusan Teknik Informatika Fakultas Teknologi Informasi di Institut Teknologi Sepuluh Nopember, Surabaya, Jawa Timur, Indonesia.

Di jurusan Teknik Informatika, penulis mengambil bidang minat Interaksi Grafis dan Seni dan memiliki kompetensi pada beberapa subjek seperti Desain Web, Pemrograman Android, Pemrograman Windows Phone, dan Big Data. Selama berada di dunia akademi kampus, penulis aktif sebagai asisten dosen untuk mata kuliah Interaksi Manusia dan Komputer. Selain itu penulis juga aktif dalam bidang nonakademik. Organisasi mahasiswa yang pernah diikuti penulis adalah menjadi Staf Departemen Dalam Negeri pada Himpunan Mahasiswa Teknik Computer-Informatika (HMTC) periode 2012-2013, Ketua Sponsorship dalan acara Schematics 2013 yang diselenggarakan oleh Himpunan Mahasiswa Teknik Computer-Informatika (HMTC) periode 2013-2014, dan menjadi peserta di berbagai kegiatan jurusan. Penulis juga pernah menjadi finalis pada beberapa ajang perlombaan seperti, VOCOMFEST tahun 2014, ENUMERATION tahun 2014, dan DISCOVERY tahun 2015. Penulis dapat dihubungi melalui alamat email risal.andika@gmail.com

BAB I
PENDAHULUAN

Pada bab ini akan dipaparkan mengenai garis besar tugas akhir yang meliputi latar belakang, tujuan, rumusan, batasan permasalahan, dan manfaat.

1.1 Latar Belakang

Bahasa isyarat adalah hal yang penting dalam komunikasi bagi orang yang menderita gangguan pendengaran dan juga merupakan cara untuk membaurkan mereka ke dalam masyarakat. Kecepatan menguasai bahasa dan kemampuan mereka berinteraksi sangat dibutuhkan. Mereka membutuhkan suatu media dan bahan pembelajaran tidak hanya berisi tentang komponen aural (berhubungan dengan telinga atau indra pendengaran) saja, namun juga secara visual karena lebih nyata daripada gerakan bibir. Ini merupakan alasan mengapa pengembangan dari bahasa isyarat dibutuhkan [1].

Mengacu pada survei yang dilakukan oleh Multi Center Study di Asia Tenggara, Indonesia berada pada posisi empat teratas dalam jumlah penderita tunarungu didalam populasi negaranya, dengan angka 4,6%, dibandingkan dengan Sri Lanka (8,8%), Myanmar (8,8%) dan India (6,3%) [2]. Pada tahun 1994, Departemen Kebudayaan dan Pendidikan merilis SIBI (Sistem Isyarat Bahasa Indonesia) dalam bentuk kamus. SIBI menjadi bahasa isyarat Indonesia yang resmi. Di dalamnya terdapat posisi jari dan gerakan tangan untuk merepresentasikan kosa kata Bahasa Indonesia. Gerakan isyarat di dalam SIBI telah diatur secara sistematis dan mengikuti konvensi. Namun, media pembelajaran yang beracu pada SIBI belum interaktif karena hanya tersedia dalam bentuk kamus atau buku saja.

Di sisi lain, teknologi informasi masih berkembang pesat di segala aspek kehidupan. Berbagai macam terobosan teknologi baru telah diciptakan oleh manusia, salah satunya perangkat Leap

Motion yang diciptakan pada sekitar tahun 2013. Dengan menggunakan Leap Motion, manusia dapat melakukan interaksi dengan komputer tanpa sentuhan sama sekali.

Leap Motion adalah sebuah alat yang berbentuk seperti perangkat USB *flashdisk*. Hasil deteksi dari alat ini dapat berupa masukan lokasi jari dan tangan dalam bentuk vektor yang bisa diproses selanjutnya oleh komputer. Dengan adanya kemampuan yang dimiliki alat Leap Motion tersebut, muncul sebuah ide untuk membuat rancang bangun modul pengenalan bahasa isyarat menggunakan teknologi Leap Motion. Bahasa isyarat yang digunakan mengacu pada Sistem Isyarat Bahasa Indonesia (SIBI).

1.2 Rumusan Permasalahan

Rumusan masalah yang diangkat dalam tugas akhir ini adalah sebagai berikut:

1. Bagaimana melakukan ekstraksi fitur?
2. Bagaimana membuat *classifier* untuk banyak kelas?
3. Bagaimana membuat aplikasi yang interaktif dan *robust*?

1.3 Batasan Permasalahan

Permasalahan yang dibahas dalam tugas akhir ini memiliki beberapa batasan, di antaranya sebagai berikut:

1. Perangkat lunak berbasis *desktop*.
2. Lingkungan pengembangan aplikasi menggunakan kakas bantu Microsoft Visual Studio.
3. Menggunakan Leap Motion SDK V.2 Skeletal Tracking.
4. Jarak antara Leap Motion Controller dengan ruang *input* tidak lebih dari 25 cm.
5. Algoritma *Neural Network* yang digunakan adalah *Backpropagation-Genetic Algorithm* (BP-GA).
6. Topologi *Neural Network* adalah *Multilayer Perceptron* dengan 1 *hidden layer*.

7. Bahasa isyarat yang digunakan mengacu pada Sistem Isyarat Bahasa Indonesia (SIBI).
8. Bahasa isyarat yang dikenali adalah huruf A sampai Z.
9. Tangan yang dipakai dalam aplikasi ini adalah tangan kanan.

1.4 Tujuan

Tujuan dari pembuatan tugas akhir ini adalah:

1. Dapat merancang bangun modul pengenalan bahasa isyarat dengan menggunakan teknologi Leap Motion.
2. Dapat mengimplementasikan algoritma *Backpropagation* dengan *Genetic Algorithm* untuk proses klasifikasi tangan dari pengguna yang ditangkap oleh Leap Motion.

1.5 Manfaat

Manfaat yang diharapkan dari pengembangan aplikasi tugas akhir ini adalah terciptanya sebuah modul pengenalan bahasa isyarat yang digunakan untuk pembelajaran bagi penderita tunarungu.

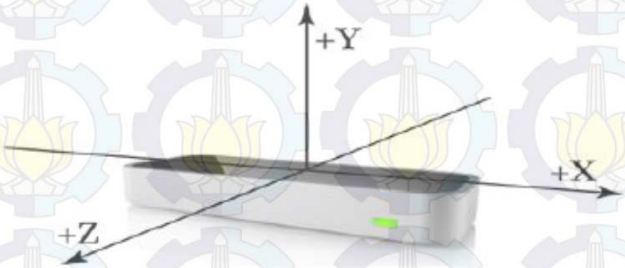
BAB II
DASAR TEORI

Pada bab ini akan dibahas mengenai dasar teori yang menjadi dasar pembuatan tugas akhir ini. Pokok permasalahan yang akan di bahas mengenai teknologi yang mendukung dalam pembuatan tugas akhir seperti Leap Motion SDK, Leap Motion Controller, *neural network*, algoritma *backpropagation*, algoritma *backpropagation-genetic*, dan pengetahuan umum mengenai bahasa isyarat huruf.

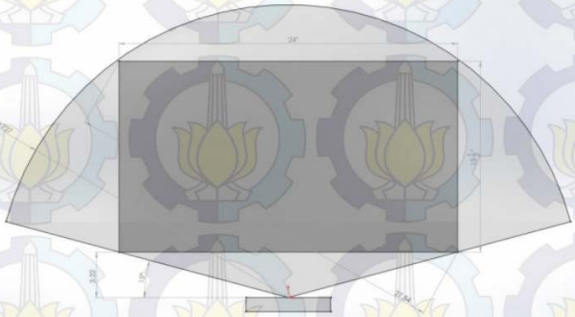
2.1 Leap Motion

Leap Motion adalah alat berukuran kecil berbasis USB yang dapat memungkinkan pengguna untuk melakukan input sebagai masukan komputer tanpa menggunakan sentuhan atau tangan sekalipun. Leap Motion merupakan perangkat keras buatan Leap Motion Inc. dari Amerika. Teknologi Leap Motion dikembangkan pertama pada tahun 2008 oleh David Holz. Pada tahun 2010, alat ini mulai dikenalkan ke publik.

Leap Motion berbentuk seperti perangkat USB *flashdisk* dengan ukuran yang lebih kecil yakni dengan berat 45 gram, panjangnya sekitar 7 sentimeter, lebarnya sekitar 1 sentimeter lebih, dan tingginya sekitar 1 sentimeter lebih seperti pada Gambar 2. 1. Alat ini dapat mendeteksi gerakan tangan dan jari manusia oleh sensor. Alat ini mempunyai 2 inframerah monokrom dan tiga infra merah jenis LED yang mampu mendeteksi area 3 dimensi tepat di atasnya [3]. Jarak deteksi maksimum alat ini sekitar 30 sentimeter dan radius deteksinya berbentuk setengah lingkaran 3 dimensi seperti pada Gambar 2. 2. Alat ini mempunyai presisi 0,7 milimeter terhadap gerakan tangan [4]. Hasil deteksi tersebut dapat berupa masukan lokasi jari dan tangan dalam bentuk vektor yang bisa diproses selanjutnya oleh komputer.



Gambar 2. 1 Leap Motion Controller



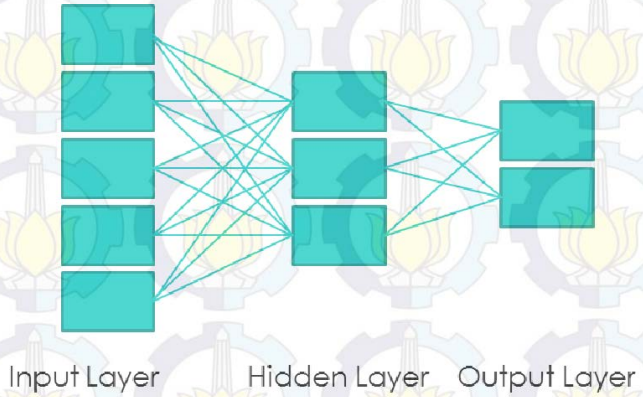
Gambar 2. 2 Radius Deteksi Leap Motion Controller

2.2 Leap Motion SDK

Leap Motion SDK ini dibuat untuk memudahkan pengembang dalam membangun aplikasi. Pengembang dapat mengunduhnya secara gratis dan tersedia dalam berbadai jenis bahasa pemrograman yang berbeda. Antara lain: Javascript, Unity / C#, C++, Java, Python, dan Objective-C. Juga mendukung berbagai macam sistem operasi yang berbeda yakni Windows, OSX, dan Linux [3].

2.3 *Neural Network*

Neural network atau jaringan syaraf neural biasanya terdiri dari 3 bagian yang disebut lapisan atau *layer* [5]. Lapisan *input* terhubung dengan lapisan *hidden* yang juga terhubung dengan lapisan *output*. *Node* yang ada pada lapisan *input* merepresentasikan fitur yang ada pada *network*. *Node* yang ada pada lapisan *hidden* didapatkan dengan aktifitas dari lapisan input dan bobot dari hubungan antara lapisan *input* dan *unit* di lapisan *hidden*. Aktifitas yang terjadi pada lapisan *hidden* tergantung dari aktiftias yang terjadi pada unit di lapisan *hidden* dan bobot antara lapisan *hidden* dan lapisan *output*. Contoh dari bentuk jaringan syaraf dengan 1 lapisan *hidden* dapat dilihat pada Gambar 2. 1.



Gambar 2. 3 Contoh Jaringan Syaraf Feedforward

2.4 *Backpropagation*

Backpropagation (BP) adalah algoritma *training* yang menggunakan *forward network* atau biasanya disebut *Multi Layer Perceptron* (MLP). Jaringan syaraf yang menggunakan algoritma BP memprediksi suatu set tertentu dengan mempelajari pada setiap

contoh yang dimasukkan ke dalam jaringan syaraf [6]. Langkah algoritma BP yaitu:

1. Mencari *error* di *node* ke-j pada lapisan *output* menggunakan rumus pada Persamaan 2.1. *Target* adalah kelas yang diinginkan.

$$Error_j = Output_j(1 - Output_j)(Target_j - Output_j) \quad (2.1)$$

2. Update bobot pada lapisan *hidden* menggunakan rumus pada Persamaan 2.2. *Wij* adalah bobot di antara lapisan i dan j (antara lapisan *hidden* dan *output*).

$$W_{ij} = W_{ij} + (Error_j * Output_i) \quad (2.2)$$

3. Mencari *error* di *node* ke-j pada lapisan *hidden* menggunakan rumus pada Persamaan 2.3. *Node* A terhubung ke *node* B dan *node* C. Error dari *node* B dan *node* C dibutuhkan untuk menghasilkan *error* pada *node* A.

$$Error_a = Output_a(1 - Output_a)(Error_b\, W_{ab} + Error_c\, W_{ac}) \quad (2.3)$$

4. Update bobot pada lapisan *input* menggunakan rumus pada Persamaan 2.2. *Wij* adalah bobot di antara lapisan i dan j (antara lapisan *input* dan *hidden*).

2.5 *Backpropagation-Genetic Algorithm Neural Network*

Sejak Rumelhalt mengenalkan *Backpropagation* (BP) pada tahun 1986, *neural network* BP telah banyak digunakan untuk data

training, dan menjadi bagian penting pada kemajuan *neural network*. Fungsi performa yang digunakan pada BP adalah MSE (*Mean Square Error*).

Rumus yang digunakan untuk mengganti berat dari jaringan pada algoritma BP standar terdapat pada Persamaan 2.4.

$$Wlij(k+1) = Wlij\,(k) + \frac{\mu\,(\partial MSE(w))}{\partial Wlji} \mid w(k) \qquad (2.4)$$

Di mana *k* adalah jumlah iterasi, *Wlij* adalah bobot yang menghubungkan *node* pada layer *i* dan *j*, μ adalah bilangan positif yang disebut *learning rate* yang digunakan untuk mengatur langkah-langkah pembelajaran dan biasanya bernilai bilangan positif sangat kecil.

Algoritma BP adalah metode yang efektif, namun mempunyai beberapa kekurangan yaitu: (1) algoritma BP menggunakan *gradient descent* untuk meminimalisir MSE dimana membuat menjadi *local minimum*; (2) Performa dari algoritma ini sangat sensitif terhadap pengaturan dari *learning rate*. Jika *learning rate* terlalu tinggi, algoritma ini mungkin menjadi tidak stabil. Jika *learning rate* terlalu kecil, algoritma ini membutuhkan waktu untuk menyatu; (3) inisialisasi berat dan bias pada jaringan sangat penting dalam proses konvergensi dari jaringan BP, di mana dihasilkan secara acak, dan terkadang membuat jaringan tidak dapan mencapai tujuan *training*.

Algoritma *Backpropagation – Genetic Algorithm Neural Network* (BPGANN) merupakan algoritma gabungan dari *Back Propagation* dan *Genetic Algorithm* (GA) dengan menggunakan metode GA untuk mencari bobot inisial dari suatu *neural network* dan mempercepat konvergensi, karena algoritma BP membuat konvergensi menjadi pelan. Langkah-langkah algoritma genetik seperti berikut:

1. Menggunakan metode GA untuk mencari kromosom paling unggul sebagai acuan bobot awal pada *neural network*.

2. Menggunakan algoritma BP untuk melatih *neural network* dengan bobot awal yang sudah ada.
3. Jika performa *neural network* semakin berbeda maka lanjut ke langkah 5, jika tidak maka lakukan langkah 4.
4. *Update* bobot dari *neural network* sebagai populasi awal dari GA dan menggunakannya untuk mencari kromosom paling unggul dari bobot yang dilatih oleh algoritma BP, lalu kembali ke langkah nomor 2.
5. Tentukan apakah kondisi pada saat iterasi memuaskan atau tidak, jika sudah memuaskan maka proses pelatihan berhenti, jika tidak maka kembali ke langkah nomor 2 [7].

2.6 Tunarungu

Tunarungu dapat diartikan sebagai keterbatasan yang dimiliki seseorang dalam mendengar sesuatu karena tidak berfungsinya organ pendengaran yang dimilikinya. Ketunarunguan dapat dibedakan menjadi dua kategori yaitu tuli (*deaf*) dan kurang dapat mendengar (*low hearing*) [8]. Tuli adalah keadaan di mana organ pendengaran telah mengalami kerusakan yang sangat parah dan mengakibatkan tidak berfungsinya pendengaran. Sedangkan kurang dapat mendengar adalah keadaan di mana organ pendengaran mengalami kerusakan tetapi masih dapat berfungsi untuk mendengar.

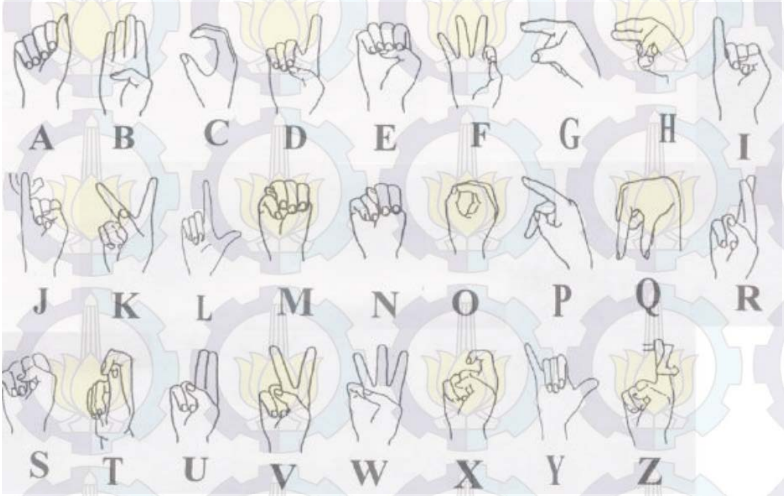
Di Indonesia, terdapat tiga macam sistem komunikasi yang digunakan oleh penderita tunarungu, yaitu:

1. Sistem Oral Murni, meliputi membaca bibir dan berbicara dengan artikulasi jelas.
2. Sistem Isyarat Murni, meliputi kosa kata isyarat, ekspresi wajah dan gerakan anggota tubuh.
3. Sistem Komunikasi Total, meliputi membaca bibir, menulis, isyarat jari, kosa kata isyarat, ekspresi wajah, dan gerakan anggota tubuh.

2.7 Bahasa Isyarat

Penderita tunarungu berkomunikasi dengan orang lain menggunakan bahasa isyarat. Bahasa isyarat berkembang dan memiliki karakteristik sendiri di setiap negara. Di Indonesia, bahasa isyarat yang digunakan mengacu pada kamus Sistem Isyarat Bahasa Indonesia atau biasa disebut dengan SIBI [8]. Jenis isyarat pada SIBI antara lain:

1. Isyarat Pokok adalah isyarat yang melambangkan sebuah kata atau konsep.
2. Isyarat Bentukan adalah isyarat yang dibentuk dengan menggabungkan isyarat pokok dengan isyarat imbuhan.
3. Abjad Jari adalah isyarat yang dibentuk dengan jari-jari untuk mengeja huruf seperti pada Gambar 2. 4.



Gambar 2. 4 Contoh Bahasa Isyarat Huruf

BAB III
ANALISIS DAN PERANCANGAN SISTEM

Bab ini membahas tahap analisis permasalahan dan perancangan dari sistem yang akan dibangun. Analisis permasalahan membahas permasalahan yang diangkat dalam pengerjaan tugas akhir. Analisis kebutuhan mencantumkan kebutuhan-kebutuhan yang diperlukan perangkat lunak. Selanjutnya dibahas mengenai perancangan sistem yang dibuat.

3.1 Analisis Perangkat Lunak

Pada subbab ini akan dibahas mengenai analisa permasalahan dari aplikasi yang akan dibuat, meliputi kebutuhan-kebutuhan perangkat lunak pada aplikasi ini. Hal yang akan dibahas meliputi deskripsi umum perangkat lunak, spesifikasi kebutuhan perangkat lunak, dan identifikasi pengguna.

3.1.1 Deskripsi Umum Perangkat Lunak

Pada tugas akhir ini akan dibangun sebuah aplikasi yang digunakan sebagai alat pengenalan bahasa isyarat yang berpacu pada SIBI (Sistem Isyarat Bahasa Indonesia). Bahasa isyarat yang dikenali adalah bahasa isyarat huruf mulai dari huruf A sampai huruf Z. Sebagai pengganti dari alat *input* perangkat keras seperti *mouse* dan *keyboard*, pada tugas akhir ini penulis akan membuat aplikasi menggunakan Leap Motion. Leap Motion merupakan sebuah alat untuk menangkap sebuah pergerakan (*motion capture*). Alat ini menangkap fitur tangan pengguna sebagai pengganti *input* ke komputer. Lalu, akan dilakukan proses klasifikasi bahasa isyarat dengan algoritma *backpropagation-genetic algorithm*.

Pembangunan aplikasi ini akan menggunakan kakas bantu Microsoft Visual Studio 2010 dengan menggunakan bahasa pemrograman C# dan Leap Motion SDK. Kemudian untuk menggunakannya, pengguna hanya menggunakan tangan sebagai

alat *input* utama, lalu Leap Motion akan mendeteksi koordinat dari tangan untuk ditampilkan ke dalam layar monitor. Arah atas, bawah, kiri, kanan yang digunakan juga sesuai antara layar monitor dan objek. Untuk menekan tombol pada layar cukup menempelkan kursor dengan objek yang ditangkap pada area cakupan Leap Motion.

3.1.2 Spesifikasi Kebutuhan Perangkat Lunak

Kebutuhan sistem yang akan dibuat ini melibatkan dua hal, yakni kebutuhan fungsional maupun kebutuhan non-fungsional. Dimana masing-masing berhubungan dengan keberhasilan dalam pembuatan aplikasi tugas akhir ini.

3.1.2.1 Kebutuhan Fungsional Perangkat Lunak

Pada sistem ini, terdapat beberapa kebutuhan fungsional yang mendukung untuk jalannya aplikasi. Fungsi yang terdapat dalam aplikasi ini adalah sebagai berikut:

a) Deteksi Gerak Objek
   Aplikasi dapat mendeteksi gerakan tangan untuk menekan tombol yang ada pada tampilan antar muka.
b) Mengekstraksi Fitur Tangan
   Aplikasi dapat mendeteksi lokasi objek berupa tangan yang akan diekstraksi menjadi fitur-fitur untuk proses klasifikasi pada saat melakukan *testing*.
c) Menerjemahkan Bahasa Isyarat
   Aplikasi dapat menerjemahkan bahasa isyarat huruf yang sesuai dengan fitur-fitur dari tangan pengguna pada saat melakukan *testing*.

3.1.2.2 Kebutuhan Non-Fungsional

Pada sistem ini, terdapat beberapa kebutuhan non-fungsional yang mendukung dan menambah performa untuk jalannya aplikasi.

Fungsi yang terdapat dalam aplikasi ini adalah sebagai berikut:

a) Penyesuaian Intensitas Cahaya

Intensitas cahaya berpengaruh dalam proses penangkapan gerakan pada Leap Motion. Semakin terang cahaya maka dapat mengganggu sensor pada Leap Motion untuk mendapatkan *input*. Sehingga, pada penggunaan aplikasi ini hendaknya tidak berada pada ruangan yang langsung terkena sinar matahari.

b) Posisi Peletakan Leap Motion

Aplikasi ini menggunakan perangkat keras Leap Motion dengan posisi menghadap pengguna. Posisi peletakan Leap Motion ini disesuaikan dengan penggunaan bahasa isyarat huruf Indonesia pada umumnya.

3.1.3 Identifikasi Pengguna

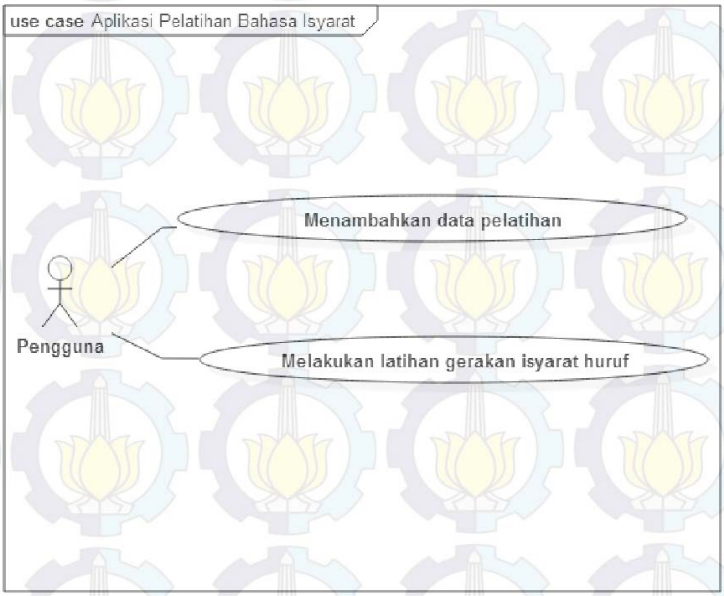
Dalam aplikasi tugas akhir ini, pengguna yang akan terlibat hanya terdapat satu orang saja, yakni orang yang akan melakukan pengenalan bahasa isyarat.

3.2 Perancangan Perangkat Lunak

Subbab ini membahas bagaimana rancangan dari aplikasi tugas akhir ini. Hal yang dibahas meliputi model kasus penggunaan, definisi aktor, definisi kasus penggunaan, arsitektur umum sistem, rancangan antarmuka aplikasi, dan rancangan proses aplikasi.

3.2.1 Model Kasus Penggunaan

Dari hasil analisa deskripsi umum perangkat lunak dan spesifikasi kebutuhan perangkat lunak yang telah dijelaskan, maka model kasus penggunaan untuk aplikasi iSyarat terlihat pada Gambar 3. 1.



Gambar 3. 1 Diagram Kasus Penggunaan Aplikasi

3.2.2 Definisi Aktor

Aktor yang terdapat dalam sistem aplikasi iSyarat terlihat pada Tabel 3. 1.

Tabel 3. 1 Definisi Kasus Penggunaan

No	Nama	Deskripsi
1	Pengguna	Merupakan aktor yang bertugas untuk menambahkan data pelatihan dan melakukan latihan gerakan isyarat huruf, seluruh fungsionalitas yang ada di dalam sistem dapat digunakan oleh pengguna.

3.2.3 Definisi Kasus Penggunaan

Pada Gambar 3. 1 telah dijelaskan bahwa aktor yang dalam hal ini disebut pengguna mempunyai dua kasus penggunaan, yakni menambahkan data pelatihan dan melakukan latihan gerakan isyarat. Detail mengenai kasus penggunaan tersebut dapat dilihat pada Tabel 3. 2.

Tabel 3. 2 Definisi Kasus Penggunaan

No	Kode Kasus Penggunaan	Nama Kasus Penggunaan	Keterangan
1	UC-01	Menambahkan Data Pelatihan	Pengguna dapat menambahkan data pelatihan huruf.
2	UC-02	Melakukan Latihan Gerakan Isyarat Huruf	Pengguna dapat melakukan latihan gerakan isyarat dengan menggunakan data pelatihan yang sebelumnya.

3.2.3.1 Kasus Penggunaan Menambahkan Data Pelatihan

Spesifikasi kasus penggunaan menambahkan data pelatihan dapat dilihat pada Tabel 3. 3.

Tabel 3. 3 Spesifikasi Kasus Penggunaan Menambahkan Data Pelatihan

Nama Kasus Penggunaan	Menambahkan Data Pelatihan
Nomor	UC-01
Deskripsi	Pengguna dapat menambahkan data pelatihan.
Aktor	Pengguna
Kondisi Awal	Pengguna berada pada halaman utama aplikasi.
Alur Normal	1. Halaman utama akan muncul sebagai tanda aplikasi telah dijalankan. 2. Pengguna mengarahkan *pointer* pada layar pada tombol *training*. 3. Pengguna mengarahkan *pointer* pada layar pada tombol mulai. 4. Pengguna menggerakkan jari sesuai dengan bahasa isyarat yang diingkan lalu mengarahkan *pointer* pada layar pada tombol berwarna merah yang terdapat di tengah layar. 5. Pengguna menunggu selama 3 detik. 6. Aplikasi akan menampilkan hasil ekstraksi fitur-fitur tangan dari pengguna. 7. Pengguna memasukkan nama huruf pada *textbox*. 8. Pengguna menekan tombol simpan untuk menyimpan hasil ekstraksi tersebut menjadi sebuah *dataset*.

	9. Pengguna memilih algoritma yang digunakan, dan menginputkan interasi *training*, lalu menekan tombol *training*.
Alur Alternatif	A1. Pengguna menekan tombol *back.* A2. Aplikasi akan kembali menampilkan halaman utama.
Kondisi Akhir	Aplikasi akan menampilkan pemberitahuan bahwa proses pelatihan telah selesai.

3.2.3.2 Kasus Penggunaan Latihan Gerakan Isyarat Huruf

Spesifikasi kasus penggunaan latihan gerakan isyarat huruf dapat dilihat pada Tabel 3. 4.

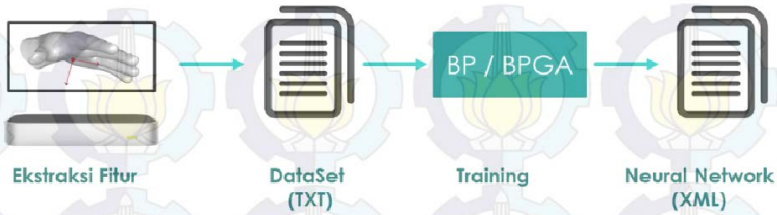
Tabel 3. 4 Spesifikasi Kasus Penggunaan Latihan Gerakan Isyarat Huruf

Nama Kasus Penggunaan	Melakukan Latihan Gerakan Isyarat Huruf
Nomor	UC-02
Deskripsi	Pengguna dapat melakukan latihan gerakan isyarat huruf.
Aktor	Pengguna
Kondisi Awal	Pengguna berada pada halaman utama aplikasi.
Alur Normal	1. Halaman utama akan muncul sebagai tanda aplikasi telah dijalankan. 2. Pengguna mengarahkan *pointer* pada layar pada tombol *testing*. 3. Pengguna mengarahkan *pointer* pada layar pada tombol mulai.

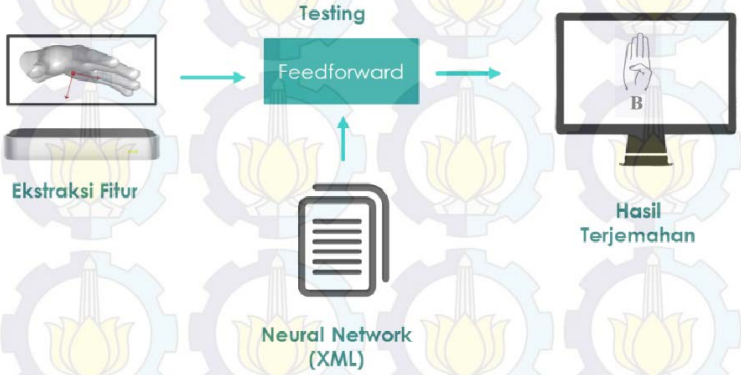
	4. Pengguna menggerakkan jari sesuai dengan bahasa isyarat yang diingkan lalu mengarahkan *pointer* pada layar pada tombol berwarna merah yang terdapat di tengah layar. 5. Pengguna menunggu selama 3 detik.
Alur Alternatif	A1. Pengguna menekan tombol *back.* A2. Aplikasi akan kembali menampilkan halaman utama.
Kondisi Akhir	Aplikasi akan melakukan proses klasifikasi berdasarkan bentuk tangan yang dibuat oleh pengguna pada saat melakukan *testing*.

3.2.4 Arsitektur Umum Sistem

Arsitektur sistem pada aplikasi yang akan dibuat terlihat pada Gambar 3. 2 dan Gambar 3. 3.



Gambar 3. 2 Arsitektur Sistem (*Training*)



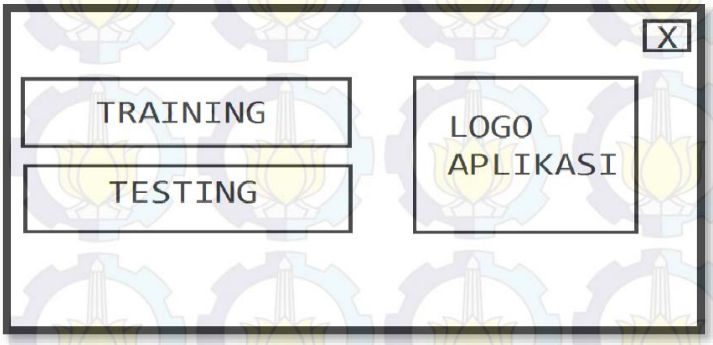
Gambar 3. 3 Arsitektur Sistem (*Testing*)

3.2.5 Rancangan Antarmuka Aplikasi

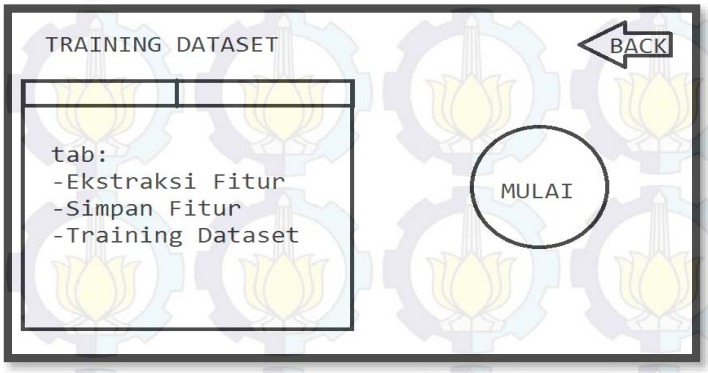
Rancangan antarmuka aplikasi diperlukan untuk memberikan gambaran umum kepada pengguna bagaimana sistem yang ada dalam aplikasi ini berinteraksi dengan pengguna. Selain itu rancangan ini juga memberikan gambaran bagi pengguna apakah tampilan yang sudah disediakan oleh aplikasi mudah untuk dipahami dan digunakan, sehingga akan muncul kesan *user experience* yang baik dan mudah.

3.2.5.1 Rancangan Antarmuka Halaman Utama

Pada halaman ini aplikasi akan menampilkan halaman utama aplikasi yang terdapat 2 tombol yaitu tombol untuk melakukan *testing* dan tombol untuk melakukan *training*. Jika pengguna menggerakkan pointer ke logo aplikasi, maka akan muncul informasi tentang aplikasi ini. Rancangan antarmuka halaman utama bisa dilihat pada Gambar 3. 4.



Gambar 3. 4 Halaman Utama

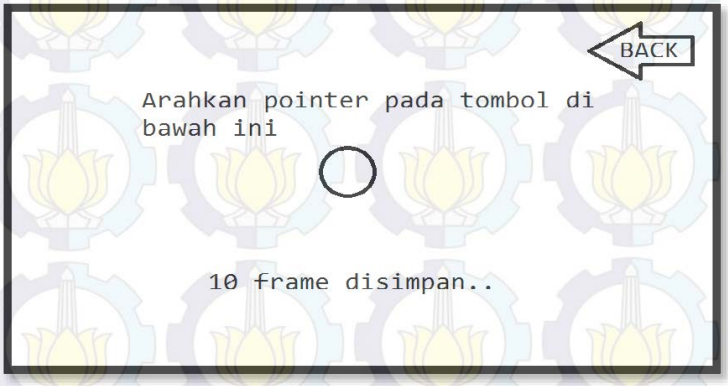


Gambar 3. 5 Halaman *Training*

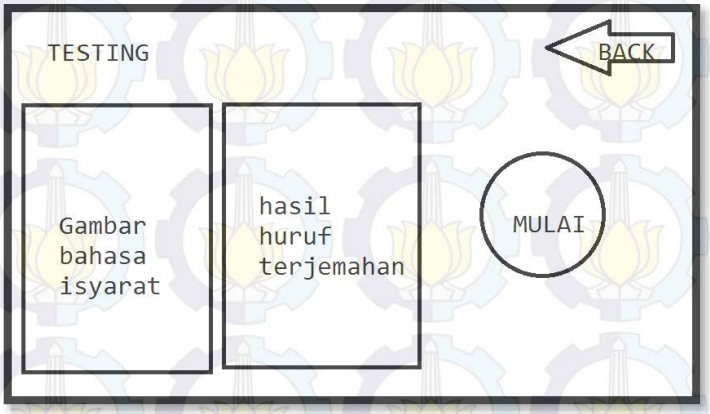
3.2.5.2 Rancangan Antarmuka *Training*

Pada halaman ini aplikasi akan menampilkan halaman *training* yang terdapat tab ekstraksi fitur hasil *training* pengguna, tombol untuk melakukan *training*, tombol untuk menyimpan hasil pelatihan huruf ke dalam bentuk *dataset* dengan ekstensi *.txt, dan tombol untuk menyimpan seluruh *dataset* menjadi sebuah *network* dengan ekstensi *.xml. Rancangan antarmuka halaman *training* bisa dilihat pada Gambar 3. 5.

Setelah pengguna menekan tombol mulai, maka akan muncul halaman pengambilan fitur tangan yang digunakan untuk *training dataset* seperti pada Gambar 3. 6. Setelah fitur disimpan, akan kembali ke halaman *training*.



Gambar 3. 6 Halaman Pengambilan Fitur Tangan Pengguna



Gambar 3. 7 Halaman *Testing*

3.2.5.3 Rancangan Antarmuka *Testing*

Pada halaman ini aplikasi akan menampilkan halaman *testing* yang terdapat hasil terjemahan dari *testing* bahasa isyarat yang dilakukan oleh pengguna. Rancangan antarmuka halaman *testing* dapat dilihat pada Gambar 3. 7.Setelah pengguna menekan tombol mulai, maka akan muncul halaman pengambilan fitur tangan yang digunakan untuk *testing* bahasa isyarat yang dilakukan oleh pengguna seperti pada Gambar 3. 6. Lalu akan dilakukan proses klasifikasi fitur tangan pengguna dan hasilnya akan ditampilkan pada halaman *testing*.

3.2.6 Rancangan Proses Aplikasi

Pada rancangan proses aplikasi akan dijelaskan mengenai proses yang terjadi dalam sistem untuk memenuhi fungsionalitas yang ada pada aplikasi. Proses ini penting agar aplikasi dapat berjalan secara baik dan benar.

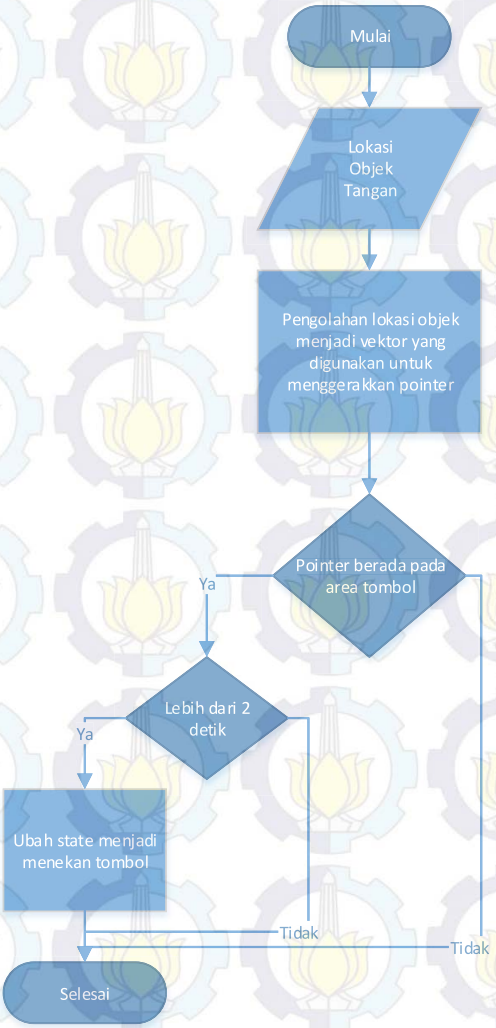
3.2.6.1 Rancangan Proses Pendeteksian Lokasi Objek

Proses ini dibutuhkan sebagai kebutuhan fungsionalitas sistem. Di mana objek yang dimaksud adalah tangan dari pengguna. Lokasi telapak tangan pengguna akan menjadi *pointer* yang dapat bergerak pada layar aplikasi. Diagram alir dapat dilihat pada Gambar 3. 8.

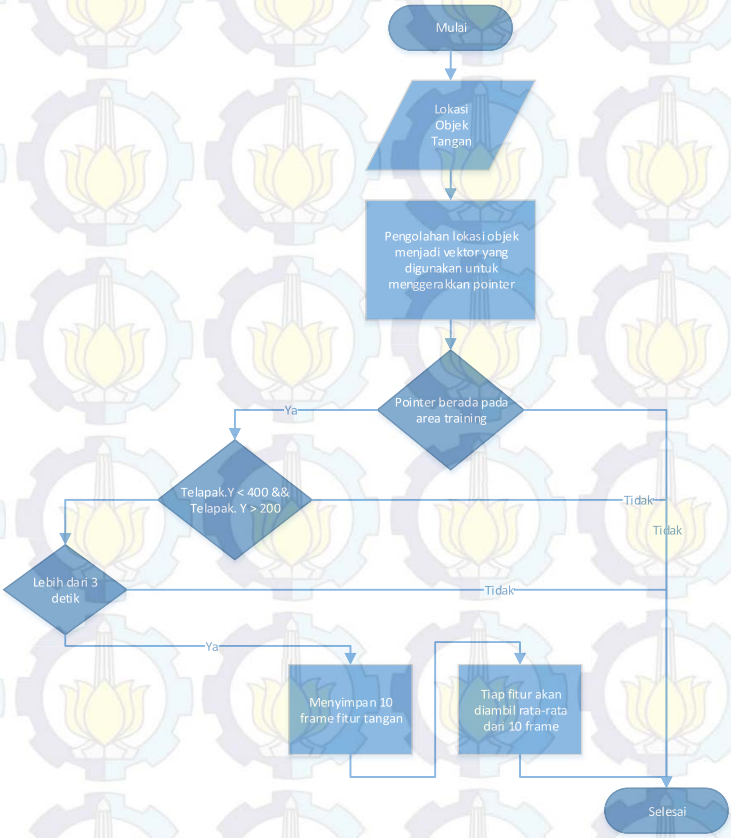
3.2.6.2 Rancangan Proses Ekstraksi Fitur

Proses ekstraksi fitur tangan sangat dibutuhkan bagi pengguna untuk melakukan *training* maupun *testing*. Fitur tangan pengguna disimpan setelah 3 detik. Leap Motion merupakan alat yang sangat sensitif, sehingga fitur tangan yang diambil

menggunakan rata-rata dari 10 *frame* yang terdeteksi. Rancangan proses pengambilan fitur tangan dapat dilihat pada Gambar 3. 9.



Gambar 3. 8 Diagram Alir Mendeteksi Lokasi Objek



Gambar 3. 9 Diagram Alir Ekstraksi Fitur Tangan Pengguna

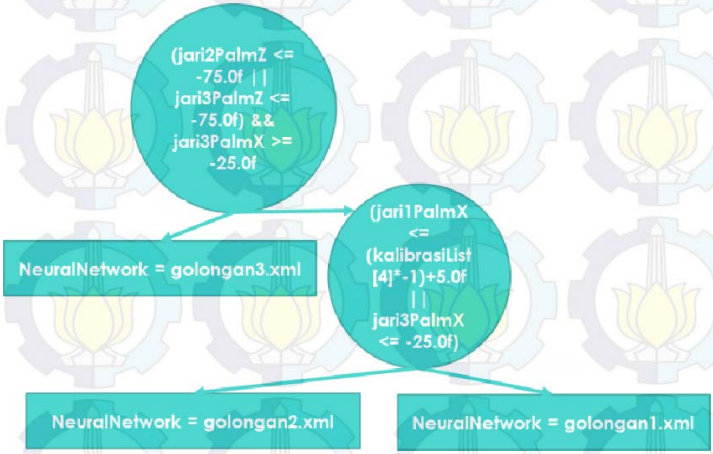
Untuk melakukan ekstraksi tangan pengguna dalam melakukan *training* maupun *testing*, penulis mengambil sebanyak 34 fitur tangan. Ibu jari disebut sebagai jari ke-1, telunjuk disebut sebagai jari ke-2, jari tengah disebut sebagai jari ke-3, jari manis disebut sebagai jari ke-4, dan jari kelingking disebut jari ke-5. Fitur ke-1 sampai dengan 4 merupakan fitur yang diambil dari *paper*. Sedangkan fitur ke-5 sampai 34 merupakan hasil pengamatan yang dilakukan oleh penulis. 34 fitur tangan yang diambil adalah:

1. Radius kepalan tangan.
2. Sudut di antara sumbu z negatif dan proyeksi dari bidang y-z.
3. Sudut di antara sumbu x negatif dan proyeksi dari bidang y-x.
4. Sudut di antara sumbu y negatif dan proyeksi dari bidang x-y.
5. Jarak jari ke-1 dengan telapak tangan terhadap sumbu x.
6. Jarak jari ke-2 dengan telapak tangan terhadap sumbu x.
7. Jarak jari ke-3 dengan telapak tangan terhadap sumbu x.
8. Jarak jari ke-4 dengan telapak tangan terhadap sumbu x.
9. Jarak jari ke-5 dengan telapak tangan terhadap sumbu x.
10. Jarak jari ke-1 dengan telapak tangan terhadap sumbu y.
11. Jarak jari ke-2 dengan telapak tangan terhadap sumbu y.
12. Jarak jari ke-3 dengan telapak tangan terhadap sumbu y.
13. Jarak jari ke-4 dengan telapak tangan terhadap sumbu y.
14. Jarak jari ke-5 dengan telapak tangan terhadap sumbu y.
15. Jarak jari ke-1 dengan telapak tangan terhadap sumbu z.
16. Jarak jari ke-2 dengan telapak tangan terhadap sumbu z.
17. Jarak jari ke-3 dengan telapak tangan terhadap sumbu z.
18. Jarak jari ke-4 dengan telapak tangan terhadap sumbu z.
19. Jarak jari ke-5 dengan telapak tangan terhadap sumbu z.
20. Jarak jari ke-2 dengan jari ke-1 terhadap sumbu x.
21. Jarak jari ke-3 dengan jari ke-1 terhadap sumbu x.
22. Jarak jari ke-4 dengan jari ke-1 terhadap sumbu x.
23. Jarak jari ke-5 dengan jari ke-1 terhadap sumbu x.
24. Jarak jari ke-2 dengan jari ke-1 terhadap sumbu y.
25. Jarak jari ke-3 dengan jari ke-1 terhadap sumbu y.
26. Jarak jari ke-4 dengan jari ke-1 terhadap sumbu y.
27. Jarak jari ke-5 dengan jari ke-1 terhadap sumbu y.
28. Jarak jari ke-2 dengan jari ke-1 terhadap sumbu z.
29. Jarak jari ke-3 dengan jari ke-1 terhadap sumbu z.
30. Jarak jari ke-4 dengan jari ke-1 terhadap sumbu z.
31. Jarak jari ke-5 dengan jari ke-1 terhadap sumbu z.
32. Jarak jari ke-2 dengan jari ke-3 terhadap sumbu x.
33. Jarak jari ke-2 dengan jari ke-3 terhadap sumbu y.
34. Jarak jari ke-2 dengan jari ke-3 terhadap sumbu z.

3.2.6.3 Rancangan Proses Pembagian *Neural Network*

Proses pembagian *neural network* pada aplikasi ini dilakukan dengan membagi 26 huruf menjadi 3 bagian. Karakteristik setiap bagian didapatkan dari hasil pengamatan penulis. Diagram pohon pembagian *neural network* dapat dilihat seperti pada Gambar 3. 10. Sedangkan *Rule* dapat dilihat pada Tabel 3. 5.

Bagian pertama untuk bahasa isyarat huruf yang letak ibu jari pada sumbu x sama atau lebih dari letak jari telunjuk pada sumbu x, yaitu huruf E, I, J, M, N, S, dan T. Bagian kedua untuk bahasa isyarat huruf yang letak ibu jari pada sumbu x lebih kecil dari letak jari telunjuk pada sumbu x, yaitu huruf A, C, D, G, H, O, P, Q, X, dan Y. Bagian ketiga yaitu untuk bahasa isyarat huruf yang jari telunjuk atau jari tengahnya tegak, yaitu huruf B, F, K, L, R, U, V, W, dan Z.



Gambar 3. 10 Diagram Pohon Pembagian *Neural Network*

Tabel 3. 5 *Rule* Pembagian *Neural Network*

Rule	*If*	*Then*
1	(jari2PalmZ <= -75.0f || jari3PalmZ <= -75.0f) && jari3PalmX >= -25.0f	Golongan3.xml
2	(jari1PalmX <= (kalibrasiList[4]*-1)+5.0f || jari3PalmX <= -25.0f)	Golongan2.xml
3	*else*	Golongan1.xml

3.2.6.4 Rancangan Proses Normalisasi Fitur

Normalisasi fitur dilakukan dengan menggunakan Persamaan 3.1. Dilakukan proses *feature scalling* untuk membuat fitur dalam *range* tertentu sehingga fitur lebih proporsional. Nilai maksimum dan minimum setiap fitur didapatkan dari hasil pengamatan yang dilakukan oleh penulis. *Pseudocode* untuk normalisasi fitur seperti pada Gambar 3. 11.

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}} \qquad (3.1)$$

Keterangan:
X = nilai fitur tangan sebelum normalisasi
X' = nilai fitur tangan setelah proses normalisasi
X_{min} = nilai fitur tangan minimum
X_{max} = nilai fitur tangan maksimum.

3.2.6.5 Rancangan Proses Kalibrasi Tangan

Kalibrasi tangan dilakukan dengan cara membandingkan tangan pengguna dengan tangan penulis menggunakan Persamaan 3.2. Setelah mendapatkan *multiplier* (nilai pengali), maka setiap fitur pengguna beracuan pada sumbu tertentu dikalikan dengan

nilai pengali. Proses ini dilakukan karena proses *training* menggunakan tangan penulis. *Pseudocode* untuk mengalibrasi telapak tangan pengguna terdapat pada Gambar 3. 12.

$$M = \frac{N_{pengguna}}{N_{penulis}} \qquad (3.2)$$

Keterangan:
M= nilai pengali
$N_{pengguna}$ = nilai lebar / nilai panjang tangan pengguna
$N_{penulis}$ = nilai lebar / nilai panjang tangan penulis.

Masukan	Fitur tangan pengguna sebelum normalisasi
Keluaran	Fitur tangan pengguna sesudah normalisasi

```
1   for i=0:FiturPengguna.Count()-1
     if i<=3
      Fitur[i] ← Fitur[i]/180
     else if i<=18
      Fitur[i] ← Fitur[i]/100
     else if i<=22
      Fitur[i] ← Fitur[i]/200
     else if i<=26
      Fitur[i] ← Fitur[i]/100
     else if i<=30
      Fitur[i] ← Fitur[i]/200
     else if i<=33
    end
```

Gambar 3. 11 *Pseudocode* Normalisasi Fitur

Masukan	Fitur tangan pengguna sebelum kalibrasi, Fitur tangan penulis yang sudah disimpan
Keluaran	Fitur tangan pengguna sesudah kalibrasi

```
3   MLebar ← (|telapakTangan.X-jariSatu.X|  +
    |telapakTangan.X-jariLima.X|) / LebarJariPenulis
4   MPanjang ← (|telapakTangan.Z-jariTiga.Z|) /
    PanjangJariPenulis
5   for i=0:FiturPengguna.Count()-1
     if i>=4 && i<=9
```

```
    FiturPengguna[i] ← FiturPengguna[i] * MLebar
  else if i>=15 && i<=19
    FiturPengguna[i] ← FiturPengguna[i] * MPanjang
  else if i>=20 && i<=22
    FiturPengguna[i] ← FiturPengguna[i] * MLebar
else if i>=27 and i<=30
    FiturPengguna[i] ← FiturPengguna[i] * MPanjang
else if i=31
    FiturPengguna[i] ← FiturPengguna[i] * MLebar
else if i=33
    FiturPengguna[i] ← FiturPengguna[i] * MPanjang
end
```

Gambar 3. 12 *Pseudocode* Kalibrasi Tangan Pengguna

BAB IV
IMPLEMENTASI

Bab ini membahas tentang implementasi dari perancangan sistem. Bab ini berisi proses implementasi dari setiap kelas pada semua modul. Namun, pada hasil akhir mungkin saja terjadi perubahan kecil. Bahasa pemrograman yang digunakan adalah bahasa pemrograman C# dengan tambahan menggunakan Leap Motion SDK.

4.1 Lingkungan Pembangunan

Dalam membangun aplikasi ini digunakan beberapa perangkat pendukung baik perangkat keras maupun perangkat lunak. Lingkungan pembangunan dijelaskan sebagai berikut.

4.1.1 Lingkungan Pembangunan Perangkat Keras

Perangkat keras yang digunakan dalam pembuatan aplikasi ini adalah sebuah laptop Lenovo G410S dengan spesifikasi sebagai berikut.

- Prosesor Intel(R) Core i5 CPU @ 2,5GHz
- Memori (RAM) 4,00 GB
- Leap Motion Controller

4.1.2 Lingkungan Pembangunan Perangkat Lunak

Spesifikasi perangkat lunak yang digunakan untuk membuat aplikasi ini sebaga berikut.

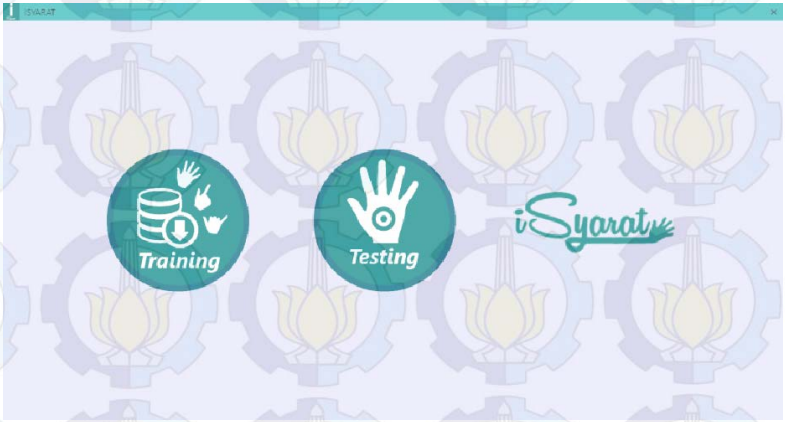
- Microsoft Visual Studio 2012
- Windows 8 64 bit sebagai sistem operasi
- Microsoft Word 2013
- Leap Motion SDK

4.2 Implementasi Antarmuka

Pada subbab ini akan dibahas mengenai hasil implementasi yang dilakukan berdasarkan rancangan antarmuka. Nantinya akan digunakan Leap Motion untuk menggerakkan kursor dalam perpindahan menu dan proses pengenalan bahasa isyarat. Dalam implementasi Antarmuka, digunakan Microsoft Visual Studio 2012 dengan bahasa pemrograman C#.

4.2.1 Implementasi Antarmuka Halaman Utama

Pada halaman utama aplikasi iSyarat ini, terdapat 2 tombol untuk menuju ke menu *training* dan menu *testing*. Cara untuk memilih tombol hanya dengan menggerakkan kursor (menggunakan Leap Motion) ke tombol yang ingin dipilih. Tampilan antarmuka halaman utama seperti pada Gambar 4. 1.

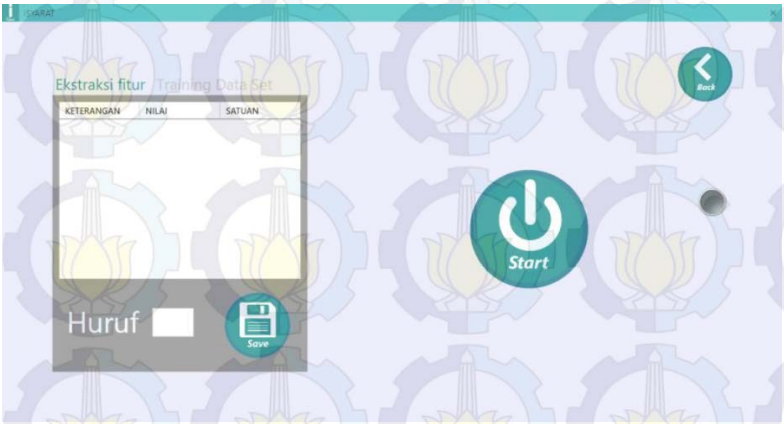


Gambar 4. 1 Antarmuka Halaman Utama

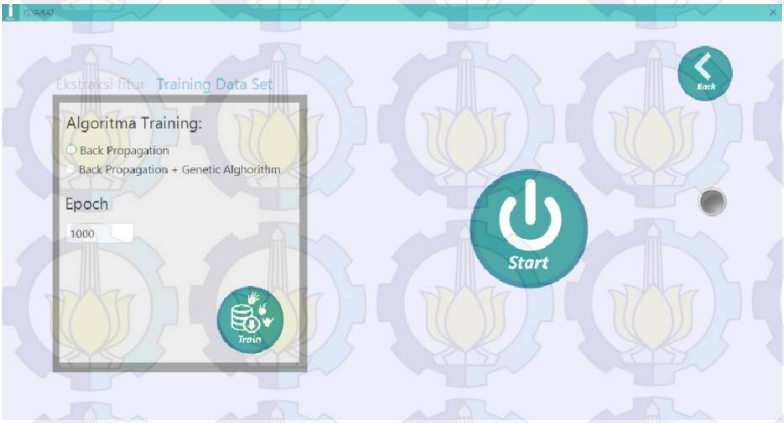
4.2.2 Implementasi Antarmuka Halaman *Training*

Pada halaman *training* ini terdapat tabel ekstraksi fitur, tombol menyimpan *dataset*, tombol untuk mulai merekam fitur

tangan, dan tombol untuk kembali ke halaman sebelumnya yaitu halaman utama seperti pada Gambar 4. 2. Ketika tab "*training dataset*" dipilih, maka akan muncul pilihan untuk melakukan *training* data seperti pilihan algoritma, dan jumlah iterasi yang dilakukan dalam menjalankan algoritma tersebut seperti pada Gambar 4. 3. Jika memilih tombol *start*, maka akan tampil antarmuka seperti pada Gambar 4. 4.



Gambar 4. 2 Antarmuka *Training* (Tabel Ekstraksi Fitur)

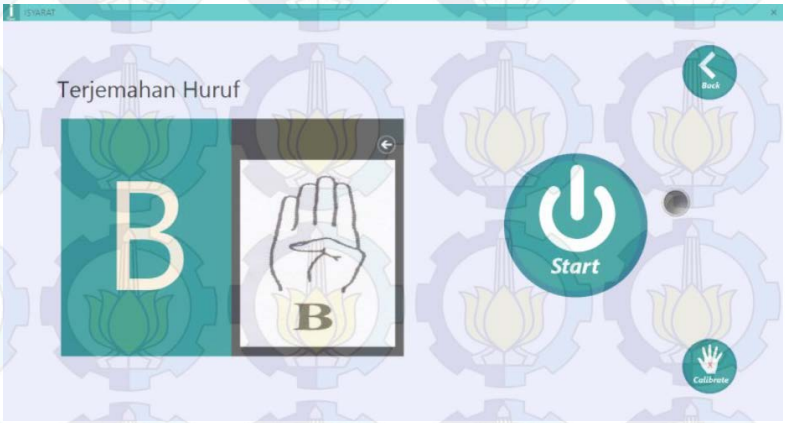


Gambar 4. 3 Antarmuka *Training* (Pilihan *Training* Data)

Gambar 4. 4 Antarmuka *Training* (Proses ekstraksi fitur)

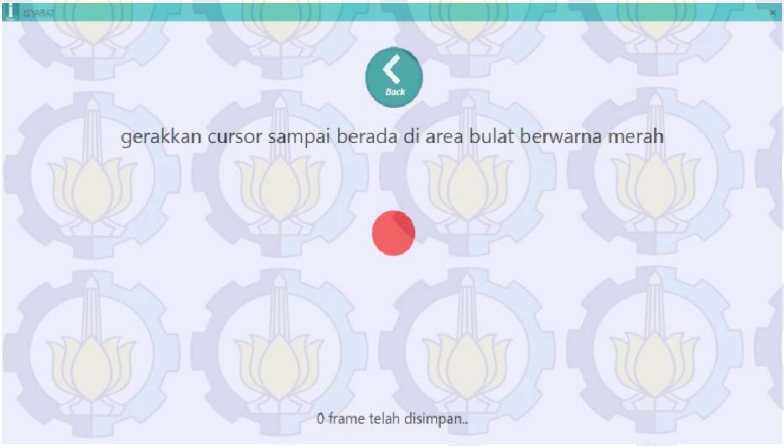
4.2.3 Implementasi Antarmuka Halaman *Testing*

Pada halaman *testing*, terdapat terjemahan dari hasil *testing* beserta gambarnya. Selain itu, juga terdapat tombol untuk memulai *testing*, tombol kembali ke menu sebelumnya, dan juga tombol untuk melakukan kalibrasi. Tampilan antarmuka halaman *testing* dapat dilihat pada Gambar 4. 5.



Gambar 4. 5 Antarmuka *Testing*

Lalu, jika memilih tombol *start*, maka akan tampil halaman pengambilan fitur jari seperti pada Gambar 4. 6. Sedangkan, jika memilih tombol kalibrasi, maka akan tampil halaman untuk kalibrasi seperti pada Gambar 4. 7.



Gambar 4. 6 Antarmuka *Testing* (Proses Ekstraksi Fitur)

Gambar 4. 7 Antarmuka Kalibrasi

4.3 Implementasi Aplikasi

Pada subbab ini akan dibahas mengenai implementasi aplikasi dari kasus penggunaan ke dalam baris kode. Dijelaskan juga dengan fungsi yang dibutuhkan untuk menunjang aplikasi ini agar dapat berjalan sebagaimana mestinya. Implementasi ini dilakukan menggunakan Microsoft Visual Studio 2012 dengan bahasa pemrograman C#.

4.3.1 Implementasi Pendeteksian Lokasi Objek

Untuk menjalankan aplikasi ini tentunya membutuhkan perangkat Leap Motion Controller, sehingga dibutuhkan suatu proses untuk mendeteksi lokasi objek tangan pengguna dan merepresentasikanya kedalam aplikasi ini sebagai kursor.

Kode sumber proses pendeteksian posisi telapak tangan kanan pengguna, dapat dilakukan seperti pada Kode Sumber 4. 1. Sementara kode sumber untuk menggerakkan kursor dapat dilakukan seperti pada Kode Sumber 4. 2

```
void newFrameHandler(Leap.Frame frame, float
deltaTime)
{
 if (frame.Hands.Count != 0){
 leapMouse.Run(frame.Hands.Rightmost.PalmPosition);
 }
}
```

Kode Sumber 4. 1 Kode Sumber Pendeteksian Telapak Tangan

```
public LeapMouse(Image img, MainWindow w){
this.img = img;
this.win = w;
}

public void Run(Vector v){
double velocity = 3.2;
```

```
img.Margin = new Thickness(v.x * velocity +
win.Width / 2, v.z * velocity + win.Height / 2, 0,
0);
}
```

Kode Sumber 4. 2. Kode Sumber Kelas *LeapMouse.cs*

4.3.2. Implementasi Proses Ekstraksi Fitur

Untuk melakukan ekstraksi tangan pengguna dalam melakukan *training* maupun *testing*, penulis mengambil sebanyak 34 fitur tangan. Pada Kode Sumber 4. 3 dijelaskan mengenai proses ekstraksi fitur pada setiap *frame* dari Leap Motion. Setiap fitur akan diambil rata-ratanya dari 10 *frame* tangan yang ditangkap oleh Leap Motion Controller.

```
handSphereRadius += hand.SphereRadius;
handPitch += Math.Abs(hand.Direction.Pitch *
(float)(180.0 / Math.PI));
handYaw += Math.Abs(hand.Direction.Yaw *
(float)(180.0 / Math.PI));
handRoll += Math.Abs(hand.Direction.Roll *
(float)(180.0 / Math.PI));

jari1PalmX += (hand.Fingers[0].TipPosition.x -
hand.PalmPosition.x);
jari2PalmX += (hand.Fingers[1].TipPosition.x -
hand.PalmPosition.x);
jari3PalmX += (hand.Fingers[2].TipPosition.x -
hand.PalmPosition.x);
jari4PalmX += (hand.Fingers[3].TipPosition.x -
hand.PalmPosition.x);
jari5PalmX += (hand.Fingers[4].TipPosition.x -
hand.PalmPosition.x);
jari1PalmY += (hand.Fingers[0].TipPosition.y -
hand.PalmPosition.y);
jari2PalmY += (hand.Fingers[1].TipPosition.y -
hand.PalmPosition.y);
jari3PalmY += (hand.Fingers[2].TipPosition.y -
hand.PalmPosition.y);
```

```
jari4PalmY += (hand.Fingers[3].TipPosition.y -
hand.PalmPosition.y);
jari5PalmY += (hand.Fingers[4].TipPosition.y -
hand.PalmPosition.y);
jari1PalmZ += (hand.Fingers[0].TipPosition.z -
hand.PalmPosition.z);
jari2PalmZ += (hand.Fingers[1].TipPosition.z -
hand.PalmPosition.z);
jari3PalmZ += (hand.Fingers[2].TipPosition.z -
hand.PalmPosition.z);
jari4PalmZ += (hand.Fingers[3].TipPosition.z -
hand.PalmPosition.z);
jari5PalmZ += (hand.Fingers[4].TipPosition.z -
hand.PalmPosition.z);

jari1ke2X += hand.Fingers[1].TipPosition.x -
hand.Fingers[0].TipPosition.x;
jari1ke3X += hand.Fingers[2].TipPosition.x -
hand.Fingers[0].TipPosition.x;
jari1ke4X += hand.Fingers[3].TipPosition.x -
hand.Fingers[0].TipPosition.x;
jari1ke5X += hand.Fingers[4].TipPosition.x -
hand.Fingers[0].TipPosition.x;
jari1ke2Y += hand.Fingers[1].TipPosition.y -
hand.Fingers[0].TipPosition.y;
jari1ke3Y += hand.Fingers[2].TipPosition.y -
hand.Fingers[0].TipPosition.y;
jari1ke4Y += hand.Fingers[3].TipPosition.y -
hand.Fingers[0].TipPosition.y;
jari1ke5Y += hand.Fingers[4].TipPosition.y -
hand.Fingers[0].TipPosition.y;
jari1ke2Z += hand.Fingers[1].TipPosition.z -
hand.Fingers[0].TipPosition.z;
jari1ke3Z += hand.Fingers[2].TipPosition.z -
hand.Fingers[0].TipPosition.z;
jari1ke4Z += hand.Fingers[3].TipPosition.z -
hand.Fingers[0].TipPosition.z;
jari1ke5Z += hand.Fingers[4].TipPosition.z -
hand.Fingers[0].TipPosition.z;
```

```
jari2ke3X += hand.Fingers[2].TipPosition.x -
hand.Fingers[1].TipPosition.x;
jari2ke3Y += hand.Fingers[2].TipPosition.y -
hand.Fingers[1].TipPosition.y;
jari2ke3Z += hand.Fingers[2].TipPosition.z -
hand.Fingers[1].TipPosition.z;
```

Kode Sumber 4. 3 Kode Sumber Ekstraksi Fitur Tangan

4.3.3 Implementasi Algoritma BPGA

Dalam implementasi algoritma *backpropagation*, setiap bobot pada *layer input* maupun bobot pada *layer hidden* akan dideklarasikan terlebih dahulu secara acak. Penulis memberikan *range* angka acak seperti pada Kode Sumber 4. 4.

```
public void InitialiseWeight()
{
 for (int i = 0; i < InputLayer.Count; i++){
 for (int j = 0; j < HiddenLayer.Count; j++){
 InputLayer[i].SetWeight(j, GetRandom() / 100.0f);
  }
 }
 for (int i = 0; i < HiddenLayer.Count; i++){
 for (int j = 0; j < OutputLayer.Count; j++){
   HiddenLayer[i].SetWeight(j, GetRandom() / 100.0f);
  }
 }
}

int GetRandom(){
 return random.Next(0, 30);
}
```

Kode Sumber 4. 4 Fungsi Inisialisasi Bobot Pada Kelas *NeuralNetwork.cs*

Setelah itu, dilakukan proses *feedforward* seperti pada Kode Sumber 4. 5. Tahap algoritma *feedforward* ini adalah:

1. Pada *layer input*, *input layer* ke-i adalah hasil ekstraksi fitur ke-i.
2. Pada *layer hidden*, *input* dari *hidden layer* ke-j dijumlahkan dengan *output* dari *input layer* ke-i (*linear*), lalu lakukan fungsi pengaktifan *sigmoid*.
3. Pada *layer output*, *input* dari *output layer* ke-j dijumlahkan dengan perkalian antara *input hidden layer* ke-i dengan bobot yang acak (*dot product*) [7].

```
public void Run(int index){
 neuralNetwork.InitaliseInput();
 DoInputLayer(index);
}

private void DoInputLayer(int index){
 for (int i = 0; i <
dataSetList[index].AttributeCount; i++){
  neuralNetwork.InputLayer[i].Input =
dataSetList[index][i];
 }
 DoHiddenLayer();
}

private void DoHiddenLayer()
{
 for (int i = 0; i < neuralNetwork.InputLayer.Count;
i++){
 for (int j = 0; j < neuralNetwork.HiddenLayer.Count;
j++){
  neuralNetwork.HiddenLayer[j].Input +=
neuralNetwork.InputLayer[i].GetOutput(j);
  }
 }
 for (int j = 0; j < neuralNetwork.HiddenLayer.Count;
j++){
 neuralNetwork.HiddenLayer[j].Input =
Sigmoid(neuralNetwork.HiddenLayer[j].Input);
  }
  DoOutputLayer();
```

```
}

private void DoOutputLayer(){
for (int i = 0; i < neuralNetwork.HiddenLayer.Count;
i++){
 for (int j = 0; j < neuralNetwork.OutputLayer.Count;
j++)
 {
 neuralNetwork.OutputLayer[j].Input +=
neuralNetwork.HiddenLayer[i].Input *
neuralNetwork.HiddenLayer[i].GetWeight(j);
  }
 }
}

private float Sigmoid(float val){
 return 1 / (1.0f + (float)Math.Exp(-val));
}
```

Kode Sumber 4. 5 Kelas Feedforward.cs

Lalu, akan dilakukan proses *backpropagation*. Tahap algoritma *backpropagation* ini adalah:

1. Menghitung *error* di *output layer* menggunakan rumus pada Persamaan 2.1. Implementasi dapat dilihat pada Kode Sumber 4. 6.
2. *Update* bobot di *hidden layer* menggunakan rumus pada Persamaan 2.2. Implementasi dapat dilihat pada Kode Sumber 4. 8.
3. Menghitung *error* di *hidden layer* menggunakan rumus pada Persamaan 2.3. Implementasi dapat dilihat pada Kode Sumber 4. 9.
4. *Update* bobot di *input layer* menggunakan rumus pada Persamaan 2.2 Implementasi dapat dilihat pada Kode Sumber 4. 10 [7].

Sedangkan algoritma genetik mempunyai tahap-tahap seperti berikut:

1. Deklarasi kromosom seperti pada Kode Sumber 4. 7.
2. Sebanyak iterasi, lakukan *backpropagation* seperti Kode Sumber 4. 13.
3. Lalu, lakukan seleksi kromosom seperti pada Kode Sumber 4. 11.
4. Lalu, lakukan penyilangan kromosom seperti pada Kode Sumber 4. 12.
5. Lalu, lakukan mutasi kromosom seperti pada Kode Sumber 4. 14 [5].

```
private float[] GetErrorOutput(int index){
  float[] outputError = new
float[classificationClass.TargetCount];
  for (int i = 0; i <
classificationClass.TargetCount; i++){
    outputError[i] =
classificationClass.GetTarget(DataSetList[index].Clas
sName)[i] - Network.OutputLayer[i].Input;
  }
  UpdateWeightHidden(outputError, index);
  return outputError;
}
```

Kode Sumber 4. 6 Fungsi GetErrorOutput Pada Kelas Backpropagation.cs

```
private void ChromosomInit()
{
  chromosoms = new List<Chromosom>(individualCount);
  for (int i = 0; i < individualCount; i++){
    chromosoms.Add(new Chromosom(GetRandomBinary()));
  }
}
```

Kode Sumber 4. 7 Fungsi ChromosomInit pada kelas GeneticAlgorithm.cs

```
private void UpdateWeightHidden(float[] outputError,
int index){
for (int i = 0; i < Network.HiddenLayer.Count; i++){
 for (int j = 0; j < Network.OutputLayer.Count; j++){
 float newWeight =
Network.HiddenLayer[i].GetWeight(j) +
(this.learningRate * outputError[j] *
Network.HiddenLayer[i].Input);
Network.HiddenLayer[i].SetWeight(j, newWeight);
  }
 }
GetErrorHidden(outputError, index);
}
```

Kode Sumber 4. 8 Fungsi UpdateWeight di *Hidden Layer* Pada Kelas Backpropagation.cs

```
private void GetErrorHidden(float[] outputError, int
index){
float[] hiddenError = new
float[Network.HiddenLayer.Count];
for (int i = 0; i < Network.HiddenLayer.Count; i++){
 float linear = 0;
 for (int j = 0; j < Network.OutputLayer.Count; j++){
 linear += outputError[j] *
Network.HiddenLayer[i].GetWeight(j);
 }
 hiddenError[i] = Network.HiddenLayer[i].Input * (1 -
Network.HiddenLayer[i].Input) * linear;
 }
 UpdateWeightInput(hiddenError, index);
}
```

Kode Sumber 4. 9 Fungsi GetErrorHidden Pada Kelas Backpropagation.cs

```
private void UpdateWeightInput(float[] hiddenError,
int index){
for (int i = 0; i < Network.InputLayer.Count; i++){
for (int j = 0; j < Network.HiddenLayer.Count; j++){
float newWeight = Network.InputLayer[i].GetWeight(j)
+ (learningRate * hiddenError[j] *
Network.InputLayer[i].Input);
Network.InputLayer[i].SetWeight(j, newWeight);
  }
 }
}
```

Kode Sumber 4. 10 Fungsi UpdateWeightInput Pada Kelas Backpropagation.cs

```
private void DoSelection(){
 chromosoms.Sort((val1, val2) =>
val1.FitnessValue.CompareTo(val2.FitnessValue));
}
```

Kode Sumber 4. 11 Fungsi DoSelection pada kelas GeneticAlgorithm.cs

```
private void DoCrossOver(){
 for (int i = 0; i < chromosoms.Count; i += 2){
 int nextIndex = i + 1;
 if (nextIndex >= chromosoms.Count)
 nextIndex = i - 1;
 int[] bit1 = chromosoms[i].Bit;
 int[] bit2 = chromosoms[nextIndex].Bit;
 for (int j = bit1.Length / 2; j < bit1.Length; j++){
  int t = bit1[j];
  bit1[j] = bit2[j];
  bit2[j] = t;
  }
 chromosoms[i].Bit = bit1;
 chromosoms[nextIndex].Bit = bit2;
 }
}
```

Kode Sumber 4. 12 Fungsi DoCrossOver pada kelas GeneticAlgorithm.cs

```
private void DoBackPropagation(){
  for (int i = 0; i < chromosoms.Count; i++){
  ListDataSet lds = new ListDataSet(listDataSet);
  for (int j = 0; j < lds.Count; j++){
  int popCount = 0;
  for (int k = 0; k < chromosoms[i].Length; k++){
   if (chromosoms[i][k] == 0){
   lds[j].RemoveBit(k - popCount);
   popCount++;
    }
   }
  }
  NeuralNetwork nn = new NeuralNetwork();
  nn.InitialiseNetwork(lds[0].AttributeCount,
lds[0].AttributeCount / 2,
classificationClass.TargetCount);
  nn.InitialiseWeight();

  BackPropagation bp = new BackPropagation();
  bp.Initialise(nn, lds, classificationClass);
  bp.Run(5000);

  FeedForward ff = new FeedForward();
  ff.Initialise(nn, lds);

  int totalCorrect = 0;
  for (int j = 0; j < lds.Count; j++){
    ff.Run(j);
       bool correct = true;
       int[] targetClass =
classificationClass.GetTarget(lds[j].ClassName);
       for (int k = 0; k <
ff.GetActualClass().Length; k++){
        if (targetClass[k] !=
ff.GetActualClass()[k])
         correct = false;
         }
        if (correct)totalCorrect++;
        }
```

```
        chromosoms[i].FitnessValue = totalCorrect /
(float)lds.Count;
        }
}
```

Kode Sumber 4. 13 Fungsi DoBackpropagation pada kelas GeneticAlgorithm.cs

```
private void DoMutation()
{
 for (int i = 0; i < chromosoms.Count; i++){
  int chanceChromoMut = GetRandom();
  if (chanceChromoMut <= GetRandom()){
   int bitIndex = GetRandom(chromosoms[i].Length);
   if (chromosoms[i][bitIndex] == 0)
chromosoms[i][bitIndex] = 1;
   else chromosoms[i][bitIndex] = 0;
  }
 }
}
```

Kode Sumber 4. 14 Fungsi DoMutation pada kelas GeneticAlgorithm.cs

4.3.4 Implementasi Proses Normalisasi Fitur

Proses normalisasi fitur mengimplementasikan *pseudocode* pada Gambar 3. 11, di mana setiap fitur pengguna dibagi dengan nilai maksimum. Nilai maksimum didapatkan penulis pada saat melakukan pengamatan pada setiap fitur. Normalisasi ini dilakukan supaya *training neural network* berjalan dengan cepat. Setiap fitur bernilai antara 0 sampai 1. Implementasi dapat dilihat pada Kode Sumber 4. 15.

4.3.5 Implementasi Proses Kalibrasi Tangan

Proses kalibrasi tangan mengimplementasikan *pseudocode* pada Gambar 3. 12. Kalibrasi tangan dilakukan karena ukuran tangan setiap pengguna berbeda-beda. Implementasi mendapatkan

perbandingan tangan penulis dan pengguna dapat dilihat pada Kode Sumber 4. 16. Sedangkan implementasi proses kalibrasi tangan pengguna dapat dilihat pada Kode Sumber 4. 17.

```
public void Normalized(List<float> kalibrasiList)
{
 for (int i = 0; i < dataSetList.Count; i++)
 {
  for (int j = 0; j < dataSetList[i].AttributeCount;
j++)
  {
  if (j == 0 || j == 1 || j == 2 || j == 3){
   dataSetList[i][j] /= 180.0f;
  }
  else if (j >= 4 && j <= 18){
   dataSetList[i][j] /= 100.0f;
  }
  else if(j>=19 && j<=22){
   dataSetList[i][j] /= 200.0f;
  }
  else if (j >= 23 && j <= 26){
   dataSetList[i][j] /= 100.0f;
  }
  else if (j >= 27 && j <= 30){
   dataSetList[i][j] /= 200.0f;
  }
  else if(j==31){
   dataSetList[i][j] /= 100.0f;
  }
  else if (j == 32){
   dataSetList[i][j] /= 100.0f;
  }
  else if (j == 33){
   dataSetList[i][j] /= 100.0f;
  }
 }
}
```

Kode Sumber 4. 15 Fungsi Normalisasi Fitur

```
if (isCalibratingOnGoing){
 pointable = frame.Pointables.Frontmost;
 hands = frame.Hands;
 hand = hands[0];
 if (countCalibrasi<10){
 kalibrasiLebarTangan +=
Math.Abs(hand.Fingers[0].TipPosition.x -
hand.PalmPosition.x) +
Math.Abs(hand.Fingers[4].TipPosition.x -
hand.PalmPosition.x);
 kalibrasiPanjangTangan +=
Math.Abs(hand.Fingers[2].TipPosition.z -
hand.PalmPosition.z);
 }
 countCalibrasi++;
 if (countCalibrasi == 10){
  kalibrasiPanjangTangan /= 10;
  kalibrasiLebarTangan /= 10;
 }
}
```

Kode Sumber 4. 16 Potongan Kode Sumber Fungsi Mendapatkan Perbandingan Untuk Kalibrasi Tangan

```
public void Calibrated(List<float> multiplier){
 for (int i = 0; i < dataSetList.Count; i++){
  for (int j = 0; j < dataSetList[i].AttributeCount;
j++){
  if (j == 0 || j == 1 || j == 2 || j == 3){
  }
  else if (j == 4 || j == 5 || j == 7 || j == 8 ||
j==9){
   dataSetList[i][j] *= multiplier[35];
  }
  else if (j == 9 || j == 10 || j == 11 || j == 12 ||
j == 13){
  }
  else if (j == 15 || j == 16 || j == 17 || j==18 ||
j==19){
```

```
    dataSetList[i][j] *= multiplier[34];
   }
   else if (j >= 19 && j <= 22){
   dataSetList[i][j] *= multiplier[35];
   }
   else if (j >= 23 && j <= 26){
   }
   else if (j >= 27 && j <= 30){
    dataSetList[i][j] *= multiplier[34];
   }
   else if (j == 31){
    dataSetList[i][j] *= multiplier[35];
   }
   else if (j == 32){
   }
   else if (j == 33){
   dataSetList[i][j] *= multiplier[34];
   }
  }
}
```

Kode Sumber 4. 17 Fungsi Kalibrasi Pada Fitur Pengguna

4.3.6 Implementasi Proses Pembagian *Neural Network*

Proses pembagian *neural network* pada aplikasi ini dilakukan dengan membagi 26 huruf menjadi 3 bagian. Bagian pertama untuk bahasa isyarat huruf yang letak ibu jari pada sumbu x sama atau lebih dari letak jari telunjuk pada sumbu x, yaitu huruf E, I, J, M, N, S, dan T. Bagian kedua untuk bahasa isyarat huruf yang letak ibu jari pada sumbu x lebih kecil dari letak jari telunjuk pada sumbu x, yaitu huruf A, C, D, G, H, O, P, Q, X, dan Y. Bagian ketiga yaitu untuk bahasa isyarat huruf yang jari telunjuk atau jari tengahnya tegak, yaitu huruf B, F, K, L, R, U, V, W, dan Z. Kode sumber untuk proses penggolongan *neural netw*ork dapat dilihat pada Kode Sumber 4. 18.

```
if ((jari2PalmZ <= -75.0f || jari3PalmZ <= -75.0f) &&
jari3PalmX >= -25.0f){
 filename = "gol3_v6_15k.xml";
 sum = 2;
}
else if ((jari1PalmX <= (kalibrasiList[4]*-1)+5.0f ||
jari3PalmX <= -25.0f)){
 filename = "gol2_v6_15k.xml";
 sum = 1;
}
else{
 filename = "gol1_v6_15k.xml";
 sum = 0;
}
```

Kode Sumber 4. 18 Pembagian *Neural Network*

BAB V
PENGUJIAN DAN EVALUASI

Bab ini membahas pengujian dan evaluasi pada aplikasi yang dikembangkan. Pengujian yang dilakukan adalah pengujian terhadap kebutuhan fungsional secara keseluruhan. Pengujian dilakukan dengan beberapa skenario. Hasil evaluasi menjabarkan tentang rangkuman hasil pengujian pada bagian akhir bab ini.

5.1. Lingkungan Pembangunan

Dalam membangun aplikasi ini digunakan beberapa perangkat pendukung baik perangkat keras maupun perangkat lunak. Perangkat keras yang digunakan dalam pembuatan aplikasi ini adalah sebuah laptop Lenovo G410S yang memiliki spesifikasi sebagai berikut.

- Prosesor Intel(R) Core i5 CPU @ 2,50GHz
- Memori (RAM) 4,00 GB
- Leap Motion

5.2. Skenario Pengujian

Skenario pengujian yang dilakukan dibagi menjadi 2 skenario A dan skenario B. Pada skenario A, *neural network* yang digunakan pada saat *testing* adalah hasil *training* dari tangan pengguna yang sama. Sedangkan skenario B, *neural network* yang digunakan pada saat *testing* berbeda dengan pada saat *training*.

1. Pengujian skenario A1 merupakan pengujian akurasi menggunakan algoritma GABP terhadap bahasa isyarat huruf yang dilakukan oleh penulis dengan menggunakan 260 dataset.
2. Pengujian skenario A2 merupakan pengujian akurasi menggunakan algoritma BP terhadap bahasa isyarat huruf

yang dilakukan oleh penulis dengan menggunakan 260 dataset. Pengujian ini menggunakan kalibrasi.

3. Pengujian skenario A3 merupakan pengujian akurasi menggunakan algoritma GABP terhadap bahasa isyarat huruf yang dilakukan oleh penulis dengan menggunakan 260 dataset. Pengujian ini menggunakan kalibrasi.
4. Pengujian skenario A4 merupakan pengujian akurasi terhadap jumlah iterasi yang dilakukan pada algoritma GABP. Pengujian ini dilakukan oleh penulis dengan menggunakan 260 dataset. Pengujian ini menggunakan kalibrasi.
5. Pengujian skenario B1 merupakan pengujian akurasi menggunakan algoritma GABP terhadap bahasa isyarat huruf yang dilakukan oleh tangan orang lain dengan menggunakan 260 dataset. Pengujian ini tanpa menggunakan kalibrasi.
6. Pengujian skenario B2 merupakan pengujian akurasi menggunakan algoritma GABP terhadap bahasa isyarat huruf yang dilakukan oleh tangan orang lain dengan menggunakan 260 dataset. Pengujian ini menggunakan kalibrasi.
7. Pengujian skenario B3 merupakan pengujian akurasi menggunakan algoritma BP terhadap bahasa isyarat huruf yang dilakukan oleh tangan orang lain dengan menggunakan 520 dataset. Pengujian ini menggunakan kalibrasi.
8. Pengujian skenario B4 merupakan pengujian akurasi menggunakan algoritma GABP terhadap bahasa isyarat huruf yang dilakukan oleh tangan orang lain dengan menggunakan 520 dataset. Pengujian ini menggunakan kalibrasi.

5.2.1 Pengujian Skenario A1 dan Analisis

Pada pengujian skenario A1 ini penulis akan melakukan uji coba terhadap akurasi jika testing dilakukan dengan algoritma genetik terlebih dahulu. Uji coba ini menggunakan *dataset* sebanyak 260 buah, semua *dataset* tersebut dilatih oleh tangan penulis. Skenario dapat dilihat pada Tabel 5. 1.

Tabel 5. 1 Skenario Pengujian A1

Nama Skenario Pengujian	Pengujian Akurasi A1
Kode	SP-A1
Algoritma	*Backpropagation-Genetic Algorithm*
Kalibrasi	Tanpa kalibrasi
Jumlah *Dataset*	260 *dataset*
Penguji	Penulis
Prosedur Pengujian	Pengguna melakukan uji coba 26 huruf isyarat sebanyak 5 kali
Hasil yang Diperoleh	Akurasi 86,9%.

Dapat dilihat jika menggunakan algoritma genetik terlebih dahulu untuk menyeleksi fitur tangan yang dipakai, akurasi bertambah sekitar 10%. Pada proses uji coba ini, fitur yang digunakan algoritma genetik pada *neural network* tangan bagian 1 (huruf E, I, J, M, N, S, dan T) adalah fitur 2, 5, 6, 9, 10, 11, 12, 16, 18, 19, 20, 25, 28, 31, 32, dan 34. Sedangkan pada bagian 2 (huruf A, C, D, G, H, O, P, Q, X, dan Y) adalah fitur 1,3,4, 6, 8, 12, 14, 15, 16, 18, 21, 22, 23, 28, 29, 30, dan 31. Sedangkan pada bagian 3 (huruf B, F, K, L, R, U, V, W, dan Z) adalah fitur 2, 3, 4, 5, 6, 7, 9, 12, 13, 14, 15, 16, 19, 21, 25, 28, 31, dan 34. Tabel perbandingan akurasi dapat dilihat pada Tabel 5. 2.

Tabel 5. 2 Tabel Perbandingan Akurasi Algoritma BP dan BPGA

Hasil Uji Coba	**Algoritma**	
	Backpropagation	**BPGA**
Benar	100	113
Salah	30	17
Total Pengujian	130	130
Akurasi	76,90%	86,90%

Menurut analisis yang dilakukan oleh penulis, jika *neural network* yang digunakan menggunakan algoritma genetik terlebih

dahulu, dengan jumlah iterasi yang sama, proses pembuatan *neural network* menggunakan algoritma genetik lebih cepat. Tabel perbandingan kecepatan pembuatan *neural network* dapat dilihat pada Tabel 5. 3.

Tabel 5. 3 Tabel Perbandingan *Runtime* Algoritma (detik)

Neural Network	Algoritma	
	Backpropagation	**BPGA**
Golongan 1	406.56	290.4
Golongan 2	539.8708	385.622
Golongan 3	589.3062	420.933

5.2.2 Pengujian Skenario A2 dan Analisis

Pengujian skenario A2 merupakan pengujian akurasi menggunakan algoritma BP terhadap bahasa isyarat huruf yang dilakukan oleh penulis dengan menggunakan 260 dataset. Pengujian ini menggunakan kalibrasi. Skenario uji coba dapat dilihat pada Tabel 5. 4. Sedangkan hasil uji coba dapat dilihat pada Tabel 5. 5. Pada skenario pengujian A2, huruf J terdeteksi sebagai huruf A sebanyak 5 kali. Hal ini terjadi karena Leap Motion mendeteksi jari kelingking (jari ke-5) sebagai ibu jari (jari ke-1) seperti pada Gambar 5. 1. Selain itu, huruf M terdeteksi sebagai huruf N sebanyak 3 kali. Hal ini terjadi dikarenakan bentuk bahasa isyarat huruf M dan N yang serupa seperti pada Gambar 5. 2. Selain itu, dengan menggunakan kalibrasi, akurasi bertambah sekitar 4%.

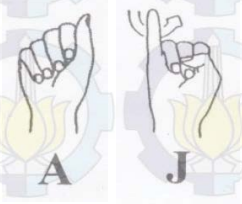
Tabel 5. 4 Skenario Pengujian A2

Nama Skenario Pengujian	Pengujian Akurasi A2
Kode	SP-A2
Algoritma	*Backpropagation*
Kalibrasi	Kalibrasi

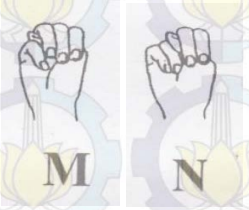
Jumlah *Dataset*	260 *dataset*
Penguji	Penulis
Prosedur Pengujian	Pengguna melakukan uji coba 26 huruf isyarat sebanyak 5 kali
Hasil yang Diperoleh	Akurasi 80,80%.

Tabel 5. 5 Terjemah Huruf Isyarat Skenario Pengujian A2

Hasil Uji Coba \ Target Kelas	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	5									5																
b		5																								
c			5																							
d				5																				2		
e					4																					
f						5																				
g							5																			
h								5																		
i									4																	
j					1					0																
k											3											1				
l												5														
m									1				1													
n													3	4					1							
o															5											
p																4	1									
q																	4									
r																		3			1					1
s													1	1					4	1						
t																				4						
u																		2			4					
v											2											4				
w																							5			
x																1								3		
y																									5	
z																										4



Gambar 5. 1 Perbandingan Huruf A dan J



Gambar 5. 2 Perbandingan Huruf M dan N

5.2.3 Pengujian Skenario A3 dan Analisis

Pengujian skenario A3 merupakan pengujian akurasi menggunakan algoritma GABP terhadap bahasa isyarat huruf yang dilakukan oleh penulis dengan menggunakan 260 dataset. Pengujian ini menggunakan kalibrasi. Pada skenario pengujian A3, akurasi terjemah bahasa isyarat naik 13% dari yang semula 80,80% menjadi 93,80%. Skenario uji coba dapat dilihat pada Tabel 5. 6. Sedangkan hasil uji coba dapat dilihat pada Tabel 5. 7.

Tabel 5. 6 Skenario Pengujian A3

Nama Skenario Pengujian	Pengujian Akurasi *Testing* A3
Kode	SP-A3
Algoritma	*Backpropagation-Genetic Algorithm*
Kalibrasi	Kalibrasi
Jumlah *Dataset*	260 *dataset*
Penguji	Penulis
Prosedur Pengujian	Pengguna melakukan uji coba 26 huruf isyarat sebanyak 5 kali
Hasil yang Diperoleh	Akurasi 93,80%.

5.2.4 Pengujian Skenario A4 dan Analisis

Pengujian skenario A4 merupakan pengujian akurasi menggunakan algoritma GABP terhadap iterasi yang dilakukan

dengan menggunakan 260 dataset. Pengujian ini menggunakan kalibrasi. Tabel perbandingan akurasi dapat dilihat pada Tabel 5. 8. Pada pengujian skenario A4 ini, didapatkan hasil iterasi yang paling optimal adalah ketika iterasi *backpropagation* yang dilakukan saat sudah mendapatkan kromosom yang baik dengan algoritma genetika adalah pada jumlah iterasi 15000.

Tabel 5. 7 Terjemah Huruf Isyarat Pengujian Skenario A3

		Target Kelas																								
		a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Hasil Uji Coba	a	5																							2		
	b		5																								
	c			5																							
	d				5												1										
	e					5																					
	f						5																				
	g							5																			
	h								5																		
	i									5																	
	j										5																
	k											5															
	l												5														
	m													4													
	n													1	4												
	o															5											
	p																4										
	q																	5									
	r																		3								1
	s														1					5							
	t																				5						
	u																		2			5					
	v																						5				
	w						1																	5			
	x																								3		
	y																									5	
	z																										4

Tabel 5. 8 Perbandingan Akurasi Terhadap Jumlah Iterasi

Jumlah Iterasi	Akurasi
10000	83,84 %
15000	93,80 %
20000	87,60 %

5.2.5 Pengujian Skenario B1 dan Analisis

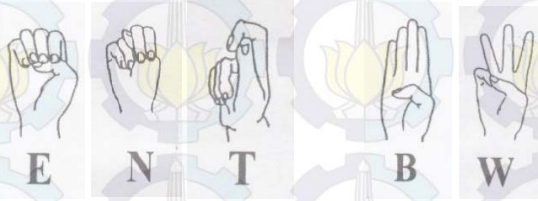
Pada pengujian skenario B1 ini, penulis akan melakukan uji coba terhadap akurasi yang didapatkan pada saat melakukan *testing* yang dilakukan oleh orang lain (panjang jari tengah dari telapak tangan adalah 91.5 mm dan lebar telapak tangan 51.6 mm) menggunakan algoritma *backpropagation-genetic algorithm*. Uji coba ini menggunakan *dataset* sebanyak 260 buah, semua *dataset* tersebut dilatih oleh tangan penulis. Skenario dapat dilihat pada Tabel 5. 9. Sedangkan hasil uji coba akurasi setiap huruf dapat dilihat seperti pada Tabel 5. 10.

Tabel 5. 9 Skenario Pengujian B1

Nama Skenario Pengujian	Pengujian Akurasi B1
Kode	SP-B1
Algoritma	*Backpropagation-Genetic Algorithm*
Kalibrasi	Tanpa kalibrasi
Jumlah *Dataset*	260 *dataset*
Penguji	Pengguna dengan panjang jari tengah 91.5 mm dan lebar telapak tangan 51.6 mm
Prosedur Pengujian	Pengguna melakukan uji coba 26 huruf isyarat sebanyak 5 kali
Hasil yang Diperoleh	Akurasi 66,9%.

Dari hasil pengujian ini, akurasi hanya mencapai 66,9%. Menurut analisis yang dilakukan oleh penulis, penyebab dari kecilnya akurasi tersebut dikarenakan tangan pengguna yang lebar dan panjang telapak tangannya lebih kecil dari tangan penulis dan tanpa melakukan kalibrasi terlebih dahulu. Huruf E terdeteksi sebagai huruf N sebanyak 3 kali juga terdeteksi sebagai huruf T sebanyak 2 kali. Huruf B terdeteksi sebagai huruf W sebanyak 3

kali. Selain bentuk tangan yang tidak sama, huruf yang dilakukan uji coba juga serupa seperti pada Gambar 5. 3.



Gambar 5. 3 Perbandingan Huruf Skenario Pengujian B1

Tabel 5. 10 Terjemah Huruf Isyarat Pengujian Skenario B1

Hasil Uji Coba	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	2																									
b		2				1																				2
c			4																							
d				4																						
e	1				0								1													
f						4																				
g							2																			
h								5																		
i									4																	
j										5									1							
k											2															
l												0														
m													1						1		2					
n					3								3	5						1						
o			1	1											5											
p																5								2		
q							3										5									
r																		5								2
s																			1							
t	2				2				1										2	4						
u																					2					
v											3											5				
w		3										5									1		5			
x																								3		
y																									5	
z																										2

5.2.6 Pengujian Skenario B2 dan Analisis

Pada pengujian skenario B2 ini, penulis akan melakukan uji coba terhadap akurasi yang didapatkan pada saat melakukan *testing* yang dilakukan oleh orang lain (panjang jari tengah dari telapak tangan adalah 91.5 mm dan lebar telapak tangan 51.6 mm) menggunakan algoritma *backpropagation-genetic algorithm*. Uji coba ini menggunakan *dataset* sebanyak 260 buah, semua *dataset* tersebut dilatih oleh tangan penulis. Uji coba ini dilakukan dengan kalibrasi terlebih dahulu. Skenario dapat dilihat pada Tabel 5. 12. Sedangkan hasil uji coba akurasi setiap huruf dapat dilihat seperti pada Tabel 5. 11.

Tabel 5. 11 Tabel Terjemah Skenario Pengujian B2

Hasil Uji Coba \ Target Kelas	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	4																									
b		3				1																				1
c			3																							
d				3																						
e	1				0								1													
f						4																				
g							3																			
h								5																		
i									3																	
j										5									1							
k											2															
l												3														
m													1						1							
n					3								3	5						1						
o			2	2											5											
p																5								2		
q							2										5									
r																		5								2
s																			1		2					
t					2				2										2	4						
u																					3					
v											3											5				
w		2										2											5			
x																								3		
y																									5	
z																										2

Tabel 5. 12 Skenario Pengujian B2

Nama Skenario Pengujian	Pengujian Akurasi B2
Kode	SP-B2
Algoritma	*Backpropagation-Genetic Algorithm*
Kalibrasi	Kalibrasi
Jumlah *Dataset*	260 *dataset*
Penguji	Pengguna dengan panjang jari tengah 91.5 mm dan lebar telapak tangan 51.6 mm
Prosedur Pengujian	Pengguna melakukan uji coba 26 huruf isyarat sebanyak 5 kali
Hasil yang Diperoleh	Akurasi 70,80%.

Pada pengujian ini, dapat dilihat bahwa pada *neural network* golongan 1 (huruf E, I, J, M, N, S) sering terjadi salah terjemah. Menurut analisa penulis, hal ini dikarenakan fitur ke 21, 22, dan 23 (jarak jari ke-1 dengan jari ke 3, 4, dan 5 pada sumbu X) diseleksi. Namun, akurasi bertambah sekitar 3,9% setelah dilakukan kalibrasi.

5.2.7 Pengujian Skenario B3 dan Evaluasi

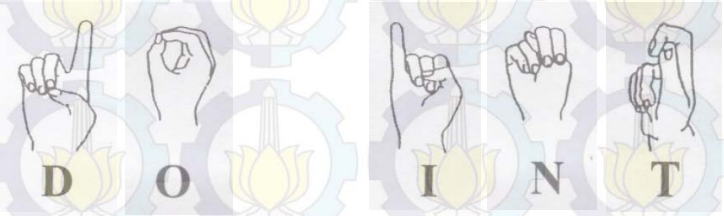
Pada pengujian skenario B3 ini, penulis akan melakukan uji coba terhadap akurasi yang didapatkan pada saat melakukan *testing*. *Testing* ini dilakukan oleh orang lain (panjang jari tengah dari telapak tangan adalah 110,56 mm dan lebar telapak tangan 74.3 mm). Uji coba ini menggunakan *dataset* sebanyak 520 buah, semua *dataset* tersebut dilatih oleh tangan penulis. Uji coba ini dilakukan dengan mengalibrasi tangan pengguna terlebih dahulu. Skenario dapat dilihat pada Tabel 5. 14. Sedangkan hasil uji coba akurasi setiap huruf dapat dilihat seperti pada Tabel 5. 13

Tabel 5. 14 Skenario Pengujian B3

Nama Skenario Pengujian	Pengujian Akurasi B3
Kode	SP-B3
Algoritma	*Backpropagation*
Kalibrasi	Kalibrasi
Jumlah *Dataset*	520 *dataset*
Penguji	Pengguna dengan panjang jari tengah 91.5 mm dan lebar telapak tangan 51.6 mm
Prosedur Pengujian	Pengguna melakukan uji coba 26 huruf isyarat sebanyak 5 kali
Hasil yang Diperoleh	Akurasi 68,5 %.

Tabel 5. 13 Tabel Terjemahan Huruf Skenario B3

Hasil Uji Coba \ Target Kelas	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	4									5																
b		5																								
c			5																							
d				1																						
e					5																					
f						5																				
g							5																			
h								1																		
i									1																	
j										0																
k											1															
l												5														
m													0							1						
n									2				5	0					2							
o	1			4				4							5									5		
p																5										
q																	5									
r																		5								
s														5					2							
t									2										1	4						
u																					5					
v											4											5				
w																							5			
x																								0		
y																									5	
z																										5



Gambar 5. 4 Perbandingan Huruf Skenario Pengujian B3

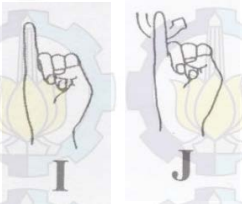
Pada skenario pengujian B3, huruf I terdeteksi sebagai huruf J sebanyak 5 kali, huruf J terdeteksi sebagai huruf I sebanyak 5 kali. Hal ini terjadi karena Leap Motion mendeteksi jari kelingking pengguna sebagai ibu jari pada saat melakukan isyarat huruf I dan J seperti pada Gambar 5. 5. Selain itu, huruf D terdeteksi sebagai huruf O sebanyak 4 kali, huruf I terdeteksi sebagai huruf N sebanyak 2 kali dan huruf T sebanyak 2 kali, huruf K terdeteksi sebagai huruf V sebanyak 4 kali, Huruf S terdeteksi sebagai huruf N sebanyak 2 kali dan huruf T sebanyak 1 kali, dan huruf X terdeteksi sebagai huruf O sebanyak 5 kali. Menurut analisis yang dilakukan oleh penulis, hal ini terjadi karena huruf yang dilakukan *testing* serupa seperti pada Gambar 5. 4.

5.2.8 Pengujian Skenario B4 dan Evaluasi

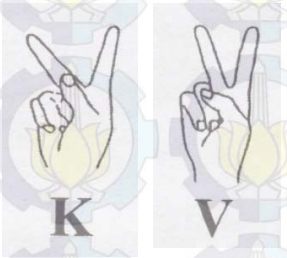
Pada pengujian skenario B4 ini penulis akan melakukan uji coba terhadap akurasi yang didapatkan pada saat melakukan *testing*. *Testing* ini dilakukan oleh orang lain (panjang jari tengah dari telapak tangan adalah 110,56 mm dan lebar telapak tangan 74.3 mm). Uji coba ini menggunakan *dataset* sebanyak 520 buah, semua *dataset* tersebut dilatih oleh tangan penulis. Uji coba ini dilakukan dengan mengalibrasi tangan pengguna terlebih dahulu. Skenario dapat dilihat pada Tabel 5. 16. Sedangkan hasil uji coba akurasi setiap huruf dapat dilihat seperti pada Tabel 5. 15.

Penguji	Orang lain dengan panjang jari tengah 110,56 mm dan lebar telapak tangan 74.3 mm
Prosedur Pengujian	Pengguna melakukan uji coba 26 huruf isyarat sebanyak 5 kali
Hasil yang Diperoleh	Akurasi 76,15%.

Pada skenario pengujian B4, huruf I terdeteksi sebagai huruf J sebanyak 5 kali, huruf J terdeteksi sebagai huruf I sebanyak 5 kali. Hal ini terjadi karena Leap Motion mendeteksi jari kelingking pengguna sebagai ibu jari pada saat melakukan isyarat huruf I dan J seperti pada Gambar 5. 5. Selain itu, huruf K terdeteksi sebagai huruf V sebanyak 4 kali. Hal ini terjadi dikarenakan bentuk bahasa isyarat huruf K dan huruf V yang serupa seperti pada Gambar 5. 6. Namun, ketika melakukan uji coba bahasa isyarat V, hasil terjemah tidak terdeteksi sebagai huruf K.



Gambar 5. 5 Perbandingan Huruf I dan J



Gambar 5. 6 Perbandingan Huruf K dan V

5.3. Evaluasi

Pada subbab ini, akan dijelaskan mengenai perbandingan dalam bentuk grafik terhadap uji coba yang telah dilakukan sebelumnya. Beberapa parameter perbandingan diantara lain:

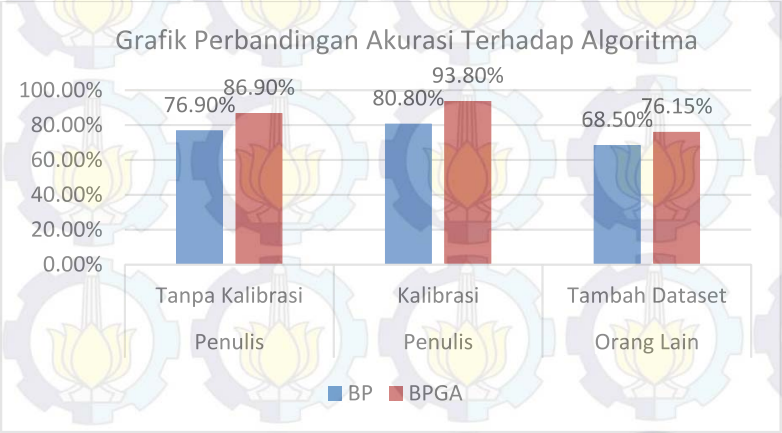
1. Akurasi model *neural network* Terhadap algoritma yang dipakai
2. Akurasi model jika ada penambahan *dataset.*
3. Akurasi model terhadap proses kalibrasi.
4. Akurasi model terhadap iterasi yang dilakukan.
5. *Runtime* model *neural network* Terhadap algoritma yang dipakai.

5.3.1 Perbandingan Akurasi Model *Neural Network* Terhadap Algoritma yang Dipakai

Pada pengujian skenario A1, selisih akurasi yang didapatkan *neural network* menggunakan algoritma genetik dan tanpa menggunakan algoritma genetik adalah sekitar 10%. Grafik perbandingan akurasi dari skenario A1 dapat dilihat seperti pada Gambar 5. 7. Selain itu, jika pengujian dilakukan kalibrasi tangan terlebih dahulu seperti pada skenario A2 dan A3, selisih akurasi sekitar 13%. Grafik perbandingan akurasi dari skenario A2 dan A3 dapat dilihat seperti pada Gambar 5. 7.

5.3.2 Perbandingan Akurasi Model Jika Ada Penambahan *Dataset*

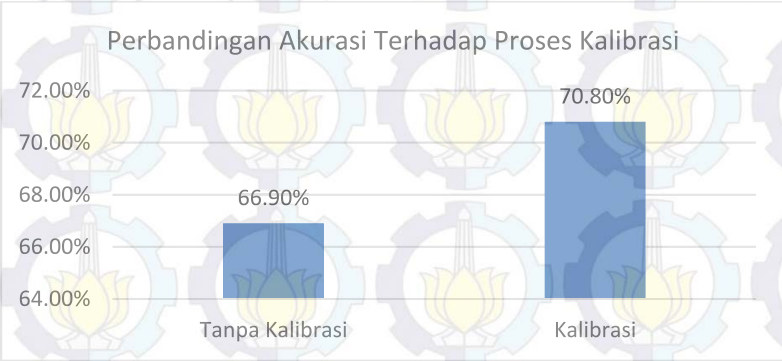
Pada pengujian skenario B3 dan B4, selisih akurasi yang didapatkan *neural network* menggunakan algoritma genetik dan tanpa menggunakan algoritma genetik adalah sekitar 7,65 %. Penambahan *dataset* dilakukan dengan tujuan untuk memperkaya model pada *neural network*. Grafik perbandingan akurasi dapat dilihat pada Gambar 5. 7.



Gambar 5. 7 Grafik Perbandingan Terhadap Algoritma

5.3.3 Perbandingan Akurasi Model Jika Menggunakan Kalibrasi

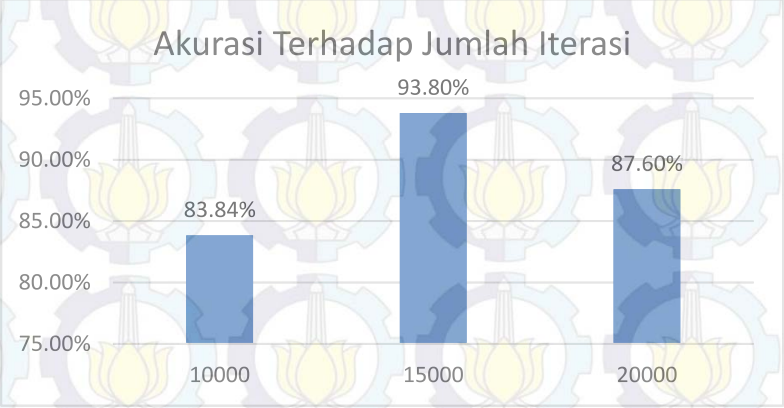
Pada pengujian skenario B1 dan B2, selisih akurasi yang didapatkan *neural network* tanpa menggunakan kalibrasi dan menggunakan kalibrasi terlebih dahulu adalah sekitar 3,9 %. Grafik perbandingan akurasi dapat dilihat pada Gambar 5. 8.



Gambar 5. 8 Grafik Perbandingan Terhadap Proses Kalibrasi

5.3.4 Perbandingan Akurasi Model Terhadap Jumlah Iterasi

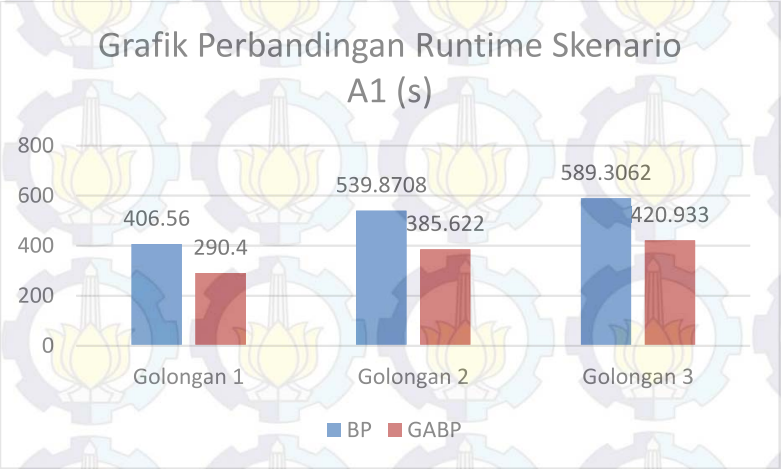
Pada pengujian skenario A4, didapatkan hasil iterasi yang paling optimal adalah ketika iterasi *backpropagation* yang dilakukan saat sudah mendapatkan kromosom yang baik dengan algoritma genetika adalah pada jumlah iterasi 15000 yaitu mencapai 93,80%. Perbandingan akurasi dapat dilihat pada Gambar 5. 9.



Gambar 5. 9 Grafik Perbandingan Akurasi Terhadap Jumlah Iterasi

5.3.5 Perbandingan *Runtime* Aplikasi Model *Neural Network* Terhadap Algoritma yang Dipakai

Pada pengujian skenario A1, *runtime* pada saat melakukan training pada *neural network* menggunakan algoritma genetika lebih cepat daripada tanpa menggunakan algoritma genetika terlebih dahulu. Grafik perbandingan *runtime* dapat dilihat pada Gambar 5. 10.



Gambar 5. 10 Grafik Perbandingan *Runtime* Skenario A1

BAB VI
KESIMPULAN DAN SARAN

Pada bab ini akan diberikan kesimpulan yang diambil selama pengerjaan tugas akhir serta saran-saran tentang pengembangan yang dapat dilakukan terhadap tugas akhir ini di masa yang akan datang.

6.1 Kesimpulan

Dari proses pengerjaan selama perancangan, implementasi, dan proses pengujian aplikasi yang dilakukan, dapat diambil kesimpulan sebagai berikut.

1. Dari sisi akurasi, penerapan algoritma GABP pada aplikasi ini lebih akurat daripada algoritma BP. Algoritma GABP dapat meningkatkan akurasi *neural network* sekitar 7-13%.
2. Dari sisi *runtime*, penerapan algoritma GABP pada aplikasi ini lebih cepat daripada algoritma BP.
3. Aplikasi yang dibangun pada tugas akhir ini dapat menerjemahkan bahasa isyarat huruf dengan akurasi di atas 80% jika pada saat *testing* menggunakan data *training* tangan yang sama. Namun, tangan yang ada pada data *training* berbeda pada saat *testing*, akurasi aplikasi ini menjadi sekitar 65-75%.
4. Leap Motion merupakan alat yang sangat sensitif terhadap gerakan, sehingga diperlukan rata-rata dalam pengambilan fitur di aplikasi ini.

6.2 Saran

Berikut saran-saran untuk pengembangan dan perbaikan sistem di masa yang akan datang. Di antaranya adalah sebagai berikut:

1. Perlu adanya penelitian lebih lanjut tentang proses kalibrasi dari jari tangan setiap pengguna. Sehingga dapat meningkatkan

nilai akurasi aplikasi ini. Hal ini harus dilakukan karena pada saat *testing* terdapat perbedaan bentuk, panjang, lebar masing-masing fitur setiap jari pengguna.

2. Memperbanyak model *neural network* dengan cara melakukan *training* dari berbagai jari pengguna.

DAFTAR PUSTAKA

[1] A. W. Yanuardi, P. Samudra and J. A. Purnama P., "Indonesian Sign Language Computer Application for The Deaf," *IEEE,* 2010.

[2] N. Ulfah, "5.000 Bayi Indonesia Lahir Tuli Setiap Tahun," Detik Health, 9 Januari 2010. [Online]. Available: http://health.detik.com/read/2010/01/09/155558/1274969/763/. [Accessed 26 Desember 2014].

[3] L. Motion, "Leap Motion," Leap Motion Inc., 2012. [Online]. Available: https://developer.leapmotion.com/documentation/csharp/index.html. [Accessed 22 Desember 2014].

[4] S. Aliyu, M. Mohandes and M. Deriche, "Arabic Sign Language Recognition using the Leap Motion Controller," *IEEE,* 2014.

[5] O. Mencer. [Online]. Available: http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html. [Accessed 6 Juni 2015].

[6] Aberdeen's Robert Gordon University, "www.rgu.ac.uk," [Online]. Available: https://www4.rgu.ac.uk/files/chapter3%20-%20bp.pdf. [Accessed 2 Maret 2015].

[7] S. Wang, S. Yin and M. Jiang, "Hybrid Neural Network Based On GA-BP for Personal Credit Scoring," *Fourth International Conference on Natural Computation,* 2008.

[8] E. Rakun, M. Andriarni, I. W. Wiprayoga, K. Danniswara and A. Tjandra, "Combining Depth Image and Skeleton Data from Kinect for Recognizing Words in the Sign System for Indonesian Language (SIBI)," *IEEE,* 2013.

BAB VI
KESIMPULAN DAN SARAN

Pada bab ini akan diberikan kesimpulan yang diambil selama pengerjaan tugas akhir serta saran-saran tentang pengembangan yang dapat dilakukan terhadap tugas akhir ini di masa yang akan datang.

6.1 Kesimpulan

Dari proses pengerjaan selama perancangan, implementasi, dan proses pengujian aplikasi yang dilakukan, dapat diambil kesimpulan sebagai berikut.

1. Dari sisi akurasi, penerapan algoritma GABP pada aplikasi ini lebih akurat daripada algoritma BP. Algoritma GABP dapat meningkatkan akurasi *neural network* sekitar 7-13%.
2. Dari sisi *runtime*, penerapan algoritma GABP pada aplikasi ini lebih cepat daripada algoritma BP.
3. Aplikasi yang dibangun pada tugas akhir ini dapat menerjemahkan bahasa isyarat huruf dengan akurasi di atas 80% jika pada saat *testing* menggunakan data *training* tangan yang sama. Namun, tangan yang ada pada data *training* berbeda pada saat *testing*, akurasi aplikasi ini menjadi sekitar 65-75%.
4. Leap Motion merupakan alat yang sangat sensitif terhadap gerakan, sehingga diperlukan rata-rata dalam pengambilan fitur di aplikasi ini.

6.2 Saran

Berikut saran-saran untuk pengembangan dan perbaikan sistem di masa yang akan datang. Di antaranya adalah sebagai berikut:

1. Perlu adanya penelitian lebih lanjut tentang proses kalibrasi dari jari tangan setiap pengguna. Sehingga dapat meningkatkan

nilai akurasi aplikasi ini. Hal ini harus dilakukan karena pada saat *testing* terdapat perbedaan bentuk, panjang, lebar masing-masing fitur setiap jari pengguna.

2. Memperbanyak model *neural network* dengan cara melakukan *training* dari berbagai jari pengguna.

LAMPIRAN A KODE SUMBER

```
class BackPropagation
{
private float learningRate = 0.03f;
private FeedForward feedForward;
private NeuralNetwork lastStableNetwork;
public NeuralNetwork Network{
 set{
  feedForward.Network = value;
 }
 get{
  return feedForward.Network;
 }
}
public ListDataSet DataSetList
{
 get{
  return feedForward.DataSetList;
 }
}
private ClassificationClass classificationClass;

public void Initialise(NeuralNetwork nn, ListDataSet
dsl, ClassificationClass cc){
 feedForward = new FeedForward();
 feedForward.Initialise(nn, dsl);
 classificationClass = cc;
}

public float Run(int maxIter)
{
 float lastAvgError = float.MaxValue;
 float avgError =0;
 for (int it = 0; it < maxIter; it++){
 avgError = 0;
 for (int k = 0; k < DataSetList.Count; k++){
  Network.InitaliseInput();
  feedForward.Run(k);
  float[] errorOutput = GetErrorOutput(k);
```

```
  float totalError = 0;
  for (int i = 0; i < errorOutput.Length; i++){
   totalError += Math.Abs(errorOutput[i]);
  }
  avgError += totalError;
 }
 avgError /= DataSetList.Count;
 }
 return avgError;
}

private float[] GetErrorOutput(int index){
  float[] outputError = new
float[classificationClass.TargetCount];
  for (int i = 0; i <
classificationClass.TargetCount; i++){
    outputError[i] =
classificationClass.GetTarget(DataSetList[index].Clas
sName)[i] - Network.OutputLayer[i].Input;
  }
  UpdateWeightHidden(outputError, index);
  return outputError;
}

private void UpdateWeightInput(float[] hiddenError, int index){
for (int i = 0; i < Network.InputLayer.Count; i++){
for (int j = 0; j < Network.HiddenLayer.Count; j++){
float newWeight = Network.InputLayer[i].GetWeight(j) + (learningRate
* hiddenError[j] * Network.InputLayer[i].Input);
Network.InputLayer[i].SetWeight(j, newWeight);
 }
 }
}
private void UpdateWeightHidden(float[] outputError,
int index){
for (int i = 0; i < Network.HiddenLayer.Count; i++){
 for (int j = 0; j < Network.OutputLayer.Count; j++){
 float newWeight =
Network.HiddenLayer[i].GetWeight(j) +
```

```
(this.learningRate * outputError[j] *
Network.HiddenLayer[i].Input);
Network.HiddenLayer[i].SetWeight(j, newWeight);
  }
 }
GetErrorHidden(outputError, index);
}

private void GetErrorHidden(float[] outputError, int
index){
float[] hiddenError = new
float[Network.HiddenLayer.Count];
for (int i = 0; i < Network.HiddenLayer.Count; i++){
 float linear = 0;
 for (int j = 0; j < Network.OutputLayer.Count; j++){
 linear += outputError[j] *
Network.HiddenLayer[i].GetWeight(j);
 }
 hiddenError[i] = Network.HiddenLayer[i].Input * (1 -
Network.HiddenLayer[i].Input) * linear;
 }
 UpdateWeightInput(hiddenError, index);
}
```

Kode Sumber 7. 1 Kelas Backpropagation.cs

```
class Chromosom
{
 private int[] bit;
 public int[] Bit{
 set{
       bit = value;
 }
 get{
  return bit;
 }
}
```

```
 public int this[int index]{
 set{
  bit[index] = value;
 }
 get{
  return bit[index];
 }
 }

 public int Length{
 get{
  return bit.Length;
 }
 }

 private float fitnessValue = 0;
 public float FitnessValue{
  set{
  fitnessValue = value;
 }
 get{
  return fitnessValue;
 }
 }

 public Chromosom(int[] newBit){
 bit = newBit;
 }

 public void Print(){
  for (int i = 0; i < bit.Length; i++){
   Console.Write(bit[i] + " ");
  }
 }
}
```

Kode Sumber 7. 2 Kelas Chromosom.cs

```
class ClassificationClass
{
private List<string> classList = new List<string>();
private int[] actualClass;
public int TargetCount
{
	get
	{
		int factor = 1;
		while (Math.Pow(2, factor) <
classList.Count)
			factor++;
		return factor;
	}
}

public void Add(string className)
{
	int index = GetIndex(className);
	if (index == -1)
	{
		classList.Add(className);
	}
}

public void Clear()
{
	classList.Clear();
}

public int GetIndex(string className)
{
	return classList.IndexOf(className);
}
public int[] GetTarget(int index)
{
	int[] target = new int[TargetCount];
	int bitCount = target.Length - 1;
```

```
        while (index > 0)
        {
                target[bitCount--] = (index % 2);
                index = index / 2;
        }

        return target;
}
public int[] GetTarget(string className)
{
        return GetTarget(GetIndex(className));
}
}
```

Kode Sumber 7. 3 Kelas ClassificationClass.cs

```
class DataSet
    {
        public string ClassName { set; get; }
        private List<float> _attribute;
        public float this[int index]
        {
            set
            {
                _attribute[index] = value;
            }
            get
            {
                return _attribute[index];
            }
        }
        public int AttributeCount
        {
            get
            {
                return _attribute.Count;
            }
        }
```

```
        public DataSet()
        {
            _attribute = new List<float>();
        }
        public DataSet(int attributeCount)
        {
            _attribute = new List<float>();
            for (int i = 0; i < attributeCount; i++)
                _attribute.Add(0);
        }
        public DataSet(DataSet ds)
        {
            _attribute = new List<float>();

            for (int i = 0; i < ds.AttributeCount;
i++)
            {
                _attribute.Add(ds[i]);
            }

            this.ClassName = ds.ClassName;
        }
        public void RemoveBit(int index)
        {
            _attribute.RemoveAt(index);
        }
    }
```

Kode Sumber 7. 4 Kelas DataSet.cs

```
class FeedForward
{
 private NeuralNetwork neuralNetwork;
 public NeuralNetwork Network
 {
 set{
 neuralNetwork = value;
 }
 get{
 return neuralNetwork;
 }
 }

 private ListDataSet dataSetList;
 public ListDataSet DataSetList
 {
 get{
  return dataSetList;
 }
 }

 public void Initialise(NeuralNetwork nn, ListDataSet
dsl){
       neuralNetwork = nn;
       dataSetList = dsl;
 }

 public void Run(int index){
       neuralNetwork.InitaliseInput();
       DoInputLayer(index);
 }

public void Initialise(NeuralNetwork nn, ListDataSet
dsl){
       neuralNetwork = nn;
       dataSetList = dsl;
 }

 public void Run(int index){
       neuralNetwork.InitaliseInput();
```

```
        DoInputLayer(index);
 }

private void DoInputLayer(int index){
 for (int i = 0; i <
dataSetList[index].AttributeCount; i++){
  neuralNetwork.InputLayer[i].Input =
dataSetList[index][i];
 }
 DoHiddenLayer();
}

private void DoHiddenLayer()
{
 for (int i = 0; i < neuralNetwork.InputLayer.Count;
i++){
 for (int j = 0; j < neuralNetwork.HiddenLayer.Count;
j++){
  neuralNetwork.HiddenLayer[j].Input +=
neuralNetwork.InputLayer[i].GetOutput(j);
  }
 }
 for (int j = 0; j < neuralNetwork.HiddenLayer.Count;
j++){
 neuralNetwork.HiddenLayer[j].Input =
Sigmoid(neuralNetwork.HiddenLayer[j].Input);
  }
  DoOutputLayer();
}

private void DoOutputLayer(){
for (int i = 0; i < neuralNetwork.HiddenLayer.Count;
i++){
 for (int j = 0; j < neuralNetwork.OutputLayer.Count;
j++)
 {
 neuralNetwork.OutputLayer[j].Input +=
neuralNetwork.HiddenLayer[i].Input *
neuralNetwork.HiddenLayer[i].GetWeight(j);
  }
```

```
 }
}

private float Sigmoid(float val){
 return 1 / (1.0f + (float)Math.Exp(-val));
}
```

Kode Sumber 7. 5 Kelas FeedForward.cs

```
class GeneticAlgorithm
{
 const int individualCount = 3;
 private Random random = new Random();
 private ClassificationClass classificationClass;
 private BackPropagation backPropagation;
 private NeuralNetwork network;
 private ListDataSet listDataSet;
 private List<Chromosom> chromosoms;

 public void Initialize(ListDataSet lds,
ClassificationClass cc){
  listDataSet = lds;
  classificationClass = cc;
 }

 public Chromosom Run(){
 ChromosomInit();
 int iteration = 3;
 for (int i = 0; i < iteration; i++){
  DoBackPropagation();
  DoSelection();
  DoCrossOver();
 int chanceMutation = GetRandom();
 if (chanceMutation < GetRandom()){
  DoMutation();
  }
 }
```

```
 int index = 0;
 for (int i = 1; i < chromosoms.Count; i++){
 if (chromosoms[i].FitnessValue >
chromosoms[index].FitnessValue){
  index = i;
  }
 }

 Chromosom fittestChromosom = chromosoms[index];
 return fittestChromosom;
 }

 private void ChromosomInit()
 {
  chromosoms = new List<Chromosom>(individualCount);
  for (int i = 0; i < individualCount; i++){
    chromosoms.Add(new Chromosom(GetRandomBinary()));
  }
 }

 private void DoSelection(){
  chromosoms.Sort((val1, val2) =>
val1.FitnessValue.CompareTo(val2.FitnessValue));
 }

 private void DoCrossOver(){
 for (int i = 0; i < chromosoms.Count; i += 2){
 int nextIndex = i + 1;
 if (nextIndex >= chromosoms.Count)
 nextIndex = i - 1;
 int[] bit1 = chromosoms[i].Bit;
 int[] bit2 = chromosoms[nextIndex].Bit;
 for (int j = bit1.Length / 2; j < bit1.Length; j++){
  int t = bit1[j];
  bit1[j] = bit2[j];
  bit2[j] = t;
  }
 chromosoms[i].Bit = bit1;
 chromosoms[nextIndex].Bit = bit2;
 }
```

```
 }

 private void DoMutation()
 {
 for (int i = 0; i < chromosoms.Count; i++){
  int chanceChromoMut = GetRandom();
  if (chanceChromoMut <= GetRandom()){
   int bitIndex = GetRandom(chromosoms[i].Length);
   if (chromosoms[i][bitIndex] == 0)
chromosoms[i][bitIndex] = 1;
   else chromosoms[i][bitIndex] = 0;
  }
 }
 }

 private void DoBackPropagation(){
  for (int i = 0; i < chromosoms.Count; i++){
  ListDataSet lds = new ListDataSet(listDataSet);
  for (int j = 0; j < lds.Count; j++){
  int popCount = 0;
  for (int k = 0; k < chromosoms[i].Length; k++){
   if (chromosoms[i][k] == 0){
   lds[j].RemoveBit(k - popCount);
   popCount++;
    }
   }
  }
  NeuralNetwork nn = new NeuralNetwork();
  nn.InitialiseNetwork(lds[0].AttributeCount,
lds[0].AttributeCount / 2,
classificationClass.TargetCount);
  nn.InitialiseWeight();

  BackPropagation bp = new BackPropagation();
  bp.Initialise(nn, lds, classificationClass);
  bp.Run(5000);

  FeedForward ff = new FeedForward();
  ff.Initialise(nn, lds);
```

```
  int totalCorrect = 0;
  for (int j = 0; j < lds.Count; j++){
    ff.Run(j);
       bool correct = true;
       int[] targetClass =
classificationClass.GetTarget(lds[j].ClassName);
       for (int k = 0; k <
ff.GetActualClass().Length; k++){
        if (targetClass[k] !=
ff.GetActualClass()[k])
         correct = false;
        }
       if (correct)totalCorrect++;
       }
       chromosoms[i].FitnessValue = totalCorrect /
(float)lds.Count;
       }
 }
}
```

Kode Sumber 7. 6 Kelas GeneticAlgorithm

```
class NeuralNetwork
{
private Random random = new Random();
public int Seed{
          set{
           random = new Random(value);
          }
}

public float treshold = 0.5f;
public List<Node> InputLayer { set; get; }
public List<Node> HiddenLayer { set; get; }
public List<Node> OutputLayer { set; get; }

public NeuralNetwork(){
}
```

```
public NeuralNetwork(NeuralNetwork newNN){
        newNN.InitaliseInput();
        InitialiseNetwork(newNN.InputLayer.Count,
newNN.HiddenLayer.Count, newNN.OutputLayer.Count);
        for (int i = 0; i < newNN.InputLayer.Count; i++){
                InputLayer[i] = newNN.InputLayer[i];
        }

        for (int i = 0; i < newNN.HiddenLayer.Count; i++){
                HiddenLayer[i] = newNN.HiddenLayer[i];
        }
}

public void InitialiseNetwork(int inputLayerCount, int
hiddenLayerCount, int outputLayerCount){
        InputLayer = new List<Node>();
        HiddenLayer = new List<Node>();
        OutputLayer = new List<Node>();

        for (int i = 0; i < inputLayerCount; i++){
                InputLayer.Add(new Node(hiddenLayerCount));
        }

        for (int i = 0; i < hiddenLayerCount; i++){
                HiddenLayer.Add(new Node(outputLayerCount));
        }

        for (int i = 0; i < outputLayerCount; i++){
                OutputLayer.Add(new Node(1));
                OutputLayer[i].SetWeight(0, 1);
        }
}

public void InitaliseInput()
{
        for (int i = 0; i < InputLayer.Count; i++){
                InputLayer[i].Input = 0;
        }

        for (int i = 0; i < HiddenLayer.Count; i++){
                HiddenLayer[i].Input = 0;
```

```
        }

        for (int i = 0; i < OutputLayer.Count; i++){
                OutputLayer[i].Input = 0;
        }
}

public void InitialiseWeight(){
        for (int i = 0; i < InputLayer.Count; i++){
                for (int j = 0; j < HiddenLayer.Count; j++){
                        InputLayer[i].SetWeight(j, GetRandom() /
100.0f);
                }
        }

        for (int i = 0; i < HiddenLayer.Count; i++){
                for (int j = 0; j < OutputLayer.Count; j++){
                        HiddenLayer[i].SetWeight(j, GetRandom() /
100.0f);
                }
        }
}

int GetRandom(){
        return random.Next(0, 30);
}
}
```

Kode Sumber 7. 7 Kelas NeuralNetwork.cs

LAMPIRAN A KODE SUMBER

```
class BackPropagation
{
private float learningRate = 0.03f;
private FeedForward feedForward;
private NeuralNetwork lastStableNetwork;
public NeuralNetwork Network{
 set{
  feedForward.Network = value;
 }
 get{
  return feedForward.Network;
 }
}
public ListDataSet DataSetList
{
 get{
  return feedForward.DataSetList;
 }
}
private ClassificationClass classificationClass;

public void Initialise(NeuralNetwork nn, ListDataSet
dsl, ClassificationClass cc){
 feedForward = new FeedForward();
 feedForward.Initialise(nn, dsl);
 classificationClass = cc;
}

public float Run(int maxIter)
{
 float lastAvgError = float.MaxValue;
 float avgError =0;
 for (int it = 0; it < maxIter; it++){
 avgError = 0;
 for (int k = 0; k < DataSetList.Count; k++){
  Network.InitaliseInput();
  feedForward.Run(k);
  float[] errorOutput = GetErrorOutput(k);
```

```
  float totalError = 0;
  for (int i = 0; i < errorOutput.Length; i++){
   totalError += Math.Abs(errorOutput[i]);
  }
  avgError += totalError;
 }
 avgError /= DataSetList.Count;
 }
 return avgError;
}

private float[] GetErrorOutput(int index){
  float[] outputError = new
float[classificationClass.TargetCount];
  for (int i = 0; i <
classificationClass.TargetCount; i++){
    outputError[i] =
classificationClass.GetTarget(DataSetList[index].Clas
sName)[i] - Network.OutputLayer[i].Input;
  }
  UpdateWeightHidden(outputError, index);
  return outputError;
}

private void UpdateWeightInput(float[] hiddenError, int index){
for (int i = 0; i < Network.InputLayer.Count; i++){
for (int j = 0; j < Network.HiddenLayer.Count; j++){
float newWeight = Network.InputLayer[i].GetWeight(j) + (learningRate
* hiddenError[j] * Network.InputLayer[i].Input);
Network.InputLayer[i].SetWeight(j, newWeight);
 }
 }
}
private void UpdateWeightHidden(float[] outputError,
int index){
for (int i = 0; i < Network.HiddenLayer.Count; i++){
 for (int j = 0; j < Network.OutputLayer.Count; j++){
 float newWeight =
Network.HiddenLayer[i].GetWeight(j) +
```

```
(this.learningRate * outputError[j] *
Network.HiddenLayer[i].Input);
Network.HiddenLayer[i].SetWeight(j, newWeight);
  }
 }
GetErrorHidden(outputError, index);
}

private void GetErrorHidden(float[] outputError, int
index){
float[] hiddenError = new
float[Network.HiddenLayer.Count];
for (int i = 0; i < Network.HiddenLayer.Count; i++){
 float linear = 0;
 for (int j = 0; j < Network.OutputLayer.Count; j++){
 linear += outputError[j] *
Network.HiddenLayer[i].GetWeight(j);
 }
 hiddenError[i] = Network.HiddenLayer[i].Input * (1 -
Network.HiddenLayer[i].Input) * linear;
 }
 UpdateWeightInput(hiddenError, index);
}
```

Kode Sumber 7. 1 Kelas Backpropagation.cs

```
class Chromosom
{
 private int[] bit;
 public int[] Bit{
 set{
       bit = value;
 }
 get{
  return bit;
 }
}
```

```
 public int this[int index]{
 set{
  bit[index] = value;
 }
 get{
  return bit[index];
 }
 }

 public int Length{
 get{
  return bit.Length;
 }
 }

 private float fitnessValue = 0;
 public float FitnessValue{
  set{
  fitnessValue = value;
 }
 get{
  return fitnessValue;
 }
 }

 public Chromosom(int[] newBit){
 bit = newBit;
 }

 public void Print(){
  for (int i = 0; i < bit.Length; i++){
   Console.Write(bit[i] + " ");
  }
 }
}
```

Kode Sumber 7. 2 Kelas Chromosom.cs

```
class ClassificationClass
{
private List<string> classList = new List<string>();
private int[] actualClass;
public int TargetCount
{
	get
	{
		int factor = 1;
		while (Math.Pow(2, factor) <
classList.Count)
			factor++;
		return factor;
	}
}

public void Add(string className)
{
	int index = GetIndex(className);
	if (index == -1)
	{
		classList.Add(className);
	}
}

public void Clear()
{
	classList.Clear();
}

public int GetIndex(string className)
{
	return classList.IndexOf(className);
}
public int[] GetTarget(int index)
{
	int[] target = new int[TargetCount];
	int bitCount = target.Length - 1;
```

```
        while (index > 0)
        {
                target[bitCount--] = (index % 2);
                index = index / 2;
        }

        return target;
}
public int[] GetTarget(string className)
{
        return GetTarget(GetIndex(className));
}
}
```

Kode Sumber 7. 3 Kelas ClassificationClass.cs

```
class DataSet
    {
        public string ClassName { set; get; }
        private List<float> _attribute;
        public float this[int index]
        {
            set
            {
                _attribute[index] = value;
            }
            get
            {
                return _attribute[index];
            }
        }
        public int AttributeCount
        {
            get
            {
                return _attribute.Count;
            }
        }
```

```
        public DataSet()
        {
            _attribute = new List<float>();
        }
        public DataSet(int attributeCount)
        {
            _attribute = new List<float>();
            for (int i = 0; i < attributeCount; i++)
                _attribute.Add(0);
        }
        public DataSet(DataSet ds)
        {
            _attribute = new List<float>();

            for (int i = 0; i < ds.AttributeCount;
i++)
            {
                _attribute.Add(ds[i]);
            }

            this.ClassName = ds.ClassName;
        }
        public void RemoveBit(int index)
        {
            _attribute.RemoveAt(index);
        }
    }
```

Kode Sumber 7. 4 Kelas DataSet.cs

```
class FeedForward
{
 private NeuralNetwork neuralNetwork;
 public NeuralNetwork Network
 {
 set{
 neuralNetwork = value;
 }
 get{
 return neuralNetwork;
 }
 }

 private ListDataSet dataSetList;
 public ListDataSet DataSetList
 {
 get{
  return dataSetList;
 }
 }

 public void Initialise(NeuralNetwork nn, ListDataSet
dsl){
       neuralNetwork = nn;
       dataSetList = dsl;
 }

 public void Run(int index){
       neuralNetwork.InitaliseInput();
       DoInputLayer(index);
 }

public void Initialise(NeuralNetwork nn, ListDataSet
dsl){
       neuralNetwork = nn;
       dataSetList = dsl;
 }

 public void Run(int index){
       neuralNetwork.InitaliseInput();
```

```
        DoInputLayer(index);
 }

private void DoInputLayer(int index){
 for (int i = 0; i <
dataSetList[index].AttributeCount; i++){
  neuralNetwork.InputLayer[i].Input =
dataSetList[index][i];
 }
 DoHiddenLayer();
}

private void DoHiddenLayer()
{
 for (int i = 0; i < neuralNetwork.InputLayer.Count;
i++){
 for (int j = 0; j < neuralNetwork.HiddenLayer.Count;
j++){
  neuralNetwork.HiddenLayer[j].Input +=
neuralNetwork.InputLayer[i].GetOutput(j);
  }
 }
 for (int j = 0; j < neuralNetwork.HiddenLayer.Count;
j++){
 neuralNetwork.HiddenLayer[j].Input =
Sigmoid(neuralNetwork.HiddenLayer[j].Input);
  }
  DoOutputLayer();
}

private void DoOutputLayer(){
for (int i = 0; i < neuralNetwork.HiddenLayer.Count;
i++){
 for (int j = 0; j < neuralNetwork.OutputLayer.Count;
j++)
 {
 neuralNetwork.OutputLayer[j].Input +=
neuralNetwork.HiddenLayer[i].Input *
neuralNetwork.HiddenLayer[i].GetWeight(j);
  }
```

```
 }
}

private float Sigmoid(float val){
 return 1 / (1.0f + (float)Math.Exp(-val));
}
```

Kode Sumber 7. 5 Kelas FeedForward.cs

```
class GeneticAlgorithm
{
 const int individualCount = 3;
 private Random random = new Random();
 private ClassificationClass classificationClass;
 private BackPropagation backPropagation;
 private NeuralNetwork network;
 private ListDataSet listDataSet;
 private List<Chromosom> chromosoms;

 public void Initialize(ListDataSet lds,
ClassificationClass cc){
  listDataSet = lds;
  classificationClass = cc;
 }

 public Chromosom Run(){
 ChromosomInit();
 int iteration = 3;
 for (int i = 0; i < iteration; i++){
  DoBackPropagation();
  DoSelection();
  DoCrossOver();
 int chanceMutation = GetRandom();
 if (chanceMutation < GetRandom()){
  DoMutation();
  }
 }
```

```
 int index = 0;
 for (int i = 1; i < chromosoms.Count; i++){
 if (chromosoms[i].FitnessValue >
chromosoms[index].FitnessValue){
  index = i;
  }
 }

 Chromosom fittestChromosom = chromosoms[index];
 return fittestChromosom;
 }

 private void ChromosomInit()
 {
  chromosoms = new List<Chromosom>(individualCount);
  for (int i = 0; i < individualCount; i++){
    chromosoms.Add(new Chromosom(GetRandomBinary()));
  }
 }

 private void DoSelection(){
  chromosoms.Sort((val1, val2) =>
val1.FitnessValue.CompareTo(val2.FitnessValue));
 }

 private void DoCrossOver(){
 for (int i = 0; i < chromosoms.Count; i += 2){
 int nextIndex = i + 1;
 if (nextIndex >= chromosoms.Count)
 nextIndex = i - 1;
 int[] bit1 = chromosoms[i].Bit;
 int[] bit2 = chromosoms[nextIndex].Bit;
 for (int j = bit1.Length / 2; j < bit1.Length; j++){
  int t = bit1[j];
  bit1[j] = bit2[j];
  bit2[j] = t;
  }
 chromosoms[i].Bit = bit1;
 chromosoms[nextIndex].Bit = bit2;
 }
```

```
 }

 private void DoMutation()
 {
 for (int i = 0; i < chromosoms.Count; i++){
  int chanceChromoMut = GetRandom();
  if (chanceChromoMut <= GetRandom()){
   int bitIndex = GetRandom(chromosoms[i].Length);
   if (chromosoms[i][bitIndex] == 0)
chromosoms[i][bitIndex] = 1;
   else chromosoms[i][bitIndex] = 0;
  }
 }
 }

 private void DoBackPropagation(){
  for (int i = 0; i < chromosoms.Count; i++){
  ListDataSet lds = new ListDataSet(listDataSet);
  for (int j = 0; j < lds.Count; j++){
  int popCount = 0;
  for (int k = 0; k < chromosoms[i].Length; k++){
   if (chromosoms[i][k] == 0){
   lds[j].RemoveBit(k - popCount);
   popCount++;
    }
   }
  }
  NeuralNetwork nn = new NeuralNetwork();
  nn.InitialiseNetwork(lds[0].AttributeCount,
lds[0].AttributeCount / 2,
classificationClass.TargetCount);
  nn.InitialiseWeight();

  BackPropagation bp = new BackPropagation();
  bp.Initialise(nn, lds, classificationClass);
  bp.Run(5000);

  FeedForward ff = new FeedForward();
  ff.Initialise(nn, lds);
```

```
  int totalCorrect = 0;
  for (int j = 0; j < lds.Count; j++){
    ff.Run(j);
       bool correct = true;
       int[] targetClass =
classificationClass.GetTarget(lds[j].ClassName);
       for (int k = 0; k <
ff.GetActualClass().Length; k++){
        if (targetClass[k] !=
ff.GetActualClass()[k])
         correct = false;
        }
       if (correct)totalCorrect++;
       }
       chromosoms[i].FitnessValue = totalCorrect /
(float)lds.Count;
       }
 }
}
```

Kode Sumber 7. 6 Kelas GeneticAlgorithm

```
class NeuralNetwork
{
private Random random = new Random();
public int Seed{
          set{
           random = new Random(value);
          }
}

public float treshold = 0.5f;
public List<Node> InputLayer { set; get; }
public List<Node> HiddenLayer { set; get; }
public List<Node> OutputLayer { set; get; }

public NeuralNetwork(){
}
```

```
public NeuralNetwork(NeuralNetwork newNN){
        newNN.InitaliseInput();
        InitialiseNetwork(newNN.InputLayer.Count,
newNN.HiddenLayer.Count, newNN.OutputLayer.Count);
        for (int i = 0; i < newNN.InputLayer.Count; i++){
                InputLayer[i] = newNN.InputLayer[i];
        }

        for (int i = 0; i < newNN.HiddenLayer.Count; i++){
                HiddenLayer[i] = newNN.HiddenLayer[i];
        }
}

public void InitialiseNetwork(int inputLayerCount, int
hiddenLayerCount, int outputLayerCount){
        InputLayer = new List<Node>();
        HiddenLayer = new List<Node>();
        OutputLayer = new List<Node>();

        for (int i = 0; i < inputLayerCount; i++){
                InputLayer.Add(new Node(hiddenLayerCount));
        }

        for (int i = 0; i < hiddenLayerCount; i++){
                HiddenLayer.Add(new Node(outputLayerCount));
        }

        for (int i = 0; i < outputLayerCount; i++){
                OutputLayer.Add(new Node(1));
                OutputLayer[i].SetWeight(0, 1);
        }
}

public void InitaliseInput()
{
        for (int i = 0; i < InputLayer.Count; i++){
                InputLayer[i].Input = 0;
        }

        for (int i = 0; i < HiddenLayer.Count; i++){
                HiddenLayer[i].Input = 0;
```

```
        }

        for (int i = 0; i < OutputLayer.Count; i++){
                OutputLayer[i].Input = 0;
        }
}

public void InitialiseWeight(){
        for (int i = 0; i < InputLayer.Count; i++){
                for (int j = 0; j < HiddenLayer.Count; j++){
                        InputLayer[i].SetWeight(j, GetRandom() /
100.0f);
                }
        }

        for (int i = 0; i < HiddenLayer.Count; i++){
                for (int j = 0; j < OutputLayer.Count; j++){
                        HiddenLayer[i].SetWeight(j, GetRandom() /
100.0f);
                }
        }
}

int GetRandom(){
        return random.Next(0, 30);
}
}
```

Kode Sumber 7. 7 Kelas NeuralNetwork.cs